6-1-2002

# Support vector machines for image and electronic mail classification

Matthew Woitaszek

# Support Vector Machines for Image and Electronic Mail Classification

by

Matthew Stephen Woitaszek

A Thesis Submitted
in
Partial Fulfillment of the
Requirements for the Degree of

Masters of Science
in
Computer Engineering

---

Dr. Muhammad Shaaban
Assistant Professor, Department of Computer Engineering

---

Dr. Andreas Savakis
Associate Professor, Department of Computer Engineering

---

Dr. Roy Czernikowski
Professor, Department of Computer Engineering

Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
June 2002

# Release Permission Form
Rochester Institute of Technology

# Support Vector Machines for Image and Electronic Mail Classification

I, Matthew Woitaszek, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part, on or after July 1, 2002. Any reproduction will not be for commercial use or profit.

June 24, 2002

Matthew Woitaszek Date

## ABSTRACT

Support Vector Machines (SVMs) have demonstrated accuracy and efficiency in a variety of binary classification applications including indoor/outdoor scene categorization of consumer photographs and distinguishing unsolicited commercial electronic mail from legitimate personal communications. This thesis examines a parallel implementation of the Sequential Minimal Optimization (SMO) method of training SVMs resulting in multiprocessor speedup subject to a decrease in accuracy dependent on the data distribution and number of processors. Subsequently the SVM classification system was applied to the image labeling and e-mail classification problems. A parallel implementation of the image classification system's color histogram, color coherence, and edge histogram feature extractors increased performance when using both non-caching and caching data distribution methods. The electronic mail classification application produced an accuracy of 96.69% with a user-generated dictionary. An implementation of the electronic mail classifier as a Microsoft Outlook add-in provides immediate mail filtering capabilities to the average desktop user. While the parallel implementation of the SVM trainer was not supported for the classification applications, the parallel feature extractor improved image classification performance.

## TABLE OF CONTENTS

**Table of Appendices** ..... **ii**
**List of Figures** ..... **iii**
**Acknowledgements** ..... **iv**
**Glossary** ..... **v**

**Introduction and Background** ..... **1**
1.1 Feature-Based SVM Classification Systems ..... 2
1.2 Other Classification Techniques ..... 3
1.3 Support Vector Machines ..... 5
1.4 Image Classification ..... 7
1.5 Electronic Mail Classification ..... 8

**Support Vector Machine Theory and Operation** ..... **11**
2.1 Support Vector Machines ..... 11
2.2 Sequential Minimal Optimization ..... 15

**Parallelization of Sequential Minimal Optimization** ..... **20**
3.1 Interleaved Parallelization ..... 25
3.2 Blocked Independent Parallelization ..... 27
3.3 Results ..... 29
3.4 Discussion ..... 31

**Parallel Image Classification** ..... **34**
4.1 Color Feature Extraction ..... 36
4.2 Texture Feature Extraction ..... 38
4.3 Combined Feature Extraction and Classification System ..... 40
4.4 Accuracy Results ..... 44
4.5 Performance Results ..... 46
4.6 Discussion ..... 47

**Electronic Mail Classification** ..... **55**
5.1 Data Sets ..... 56
5.2 Method ..... 57
5.3 Results ..... 58
5.4 Discussion ..... 59
5.5 Microsoft Outlook Integrated Spam Scanner ..... 62

**Conclusions** ..... **64**
6.1 Summary of Work ..... 64
6.2 Challenges Encountered ..... 65
6.3 Future Work ..... 66

**Bibliography** ..... **69**

## TABLE OF APPENDICES

Appendix A – SVM Software Operation ..... 71
Appendix B – Parallel SVM Results ..... 73

Appendix C – Full Image Individual Classifier Results ..... 74
Appendix D – Full Image Combined Classifier Results ..... 77
Appendix E – Subblocked Image Individual Classifier Results ..... 79
Appendix F – Subblocked Image Combined Classifier Results ..... 80
Appendix G – Image Feature Extraction Performance ..... 81
Appendix H – Image Classification Output Example ..... 82

Appendix I – Mail Classification Test Sets ..... 84
Appendix J – Mail Classification Results ..... 85
Appendix K – Mail Classification Trained Dictionary Model ..... 86
Appendix L – Mail Classification Interface ..... 89
Appendix M – Mail Classification Outlook Integration ..... 90

Appendix N – CD-ROM Content List ..... 91

## LIST OF FIGURES

### Figures

Figure 1.1 – SVM data flow block diagram ..... 2
Figure 2.1 – Separating hyperplanes in a linear SVM ..... 12
Figure 2.2 – Constraints for optimizing two Lagrange multipliers ..... 17
Figure 3.1 – Optimization pattern for two way parallel interleaved approach ..... 26
Figure 3.2 – Optimization pattern for three way parallel interleaved approach ..... 26
Figure 3.3 – SVM recall for Adult Linear ..... 30
Figure 3.4 – SVM performance for Adult Linear ..... 30
Figure 3.5 – SVM recall for Adult RBF ..... 30
Figure 3.6 – SVM performance for Adult RBF ..... 30
Figure 4.1 – Pixel enumeration for 3x3 grid Sobel operator ..... 39
Figure 4.2 – Two stage classification approach ..... 41
Figure 4.3 – Execution trace before parallelization ..... 42
Figure 4.4 – Execution trace after parallelization ..... 52

### Tables

Table 4.1 – Full image individual CH classifier results for 4-way test ..... 37
Table 4.2 – Full image individual CC classifier results for 4-way test ..... 38
Table 4.3 – Full image individual EH classifier results for 4-way test ..... 40
Table 4.4 – Parallel feature extractor processor assignments ..... 42
Table 4.5 – Final full image individual classifier accuracy ..... 45
Table 4.6 – Final full image combined classifier accuracy ..... 45
Table 4.7 – Final subblocked image individual classifier accuracy ..... 45
Table 4.8 – Final subblocked image combined classifier accuracy ..... 45
Table 4.9 – Feature extraction performance ..... 46
Table 4.10 – Image classification performance ..... 47
Table 4.11 – Number of SVs in feature classifier models ..... 53
Table 5.1 – Mail Classification Accuracy ..... 59
Table 5.2 – Trained Dictionary Statistics ..... 60

## ACKNOWLEDGEMENTS

*Research without indebtedness is suspect, and somebody must always, somehow, be thanked.* –Umberto Eco, How to Write an Introduction

I would like to take this opportunity to thank several individuals for their contributions to this thesis. First and foremost is my thesis advisor, Dr. Muhammad Shaaban, who introduced me to parallel computing and was influential in the development of this project as well as my selection of future graduate work. Dr. Roy Czernikowski and Dr. Andreas Savakis consistently applied their high standards of academic rigor, ensuring that my work was completed on time, under budget, and done right.

Several other people provided fantastic assistance making portions of this work possible. I must thank Navid Serrano of Kodak for substantial support in understanding the applications of support vector machines to image processing and Sue Muller for a walkthrough of her feature extraction thesis project. Shawn Thomas, of RIT Information Technology Services, provided the unsolicited commercial electronic mail for the spam filter training. Within the Department of Computer Engineering, Rick Tolleson and Paul Mezzanini provided assistance with computational resources. Finally, I am especially thankful for the support, encouragement, and proofreading abilities of my mother, Lynn Macken.

# GLOSSARY

**CC – Color Coherence.** A feature vector for image classification produced by generating a color histogram including only points surrounded by a region of colors within a specified tolerance (p. 37).

**CH – Color Histogram.** A feature vector for image classification describing the quantities of constituent colors in an image (p. 36).

**EH – Edge Histogram.** A feature vector for image classification produced by generating an angle histogram after the Sobel edge-detection transform. The edge histogram produces texture-related image data (p. 40).

**KKT – Karush-Kuhn-Tucker conditions.** Necessary and sufficient conditions for the termination of a SVM training process with a valid solution (p. 14).

**k-NN – k-Nearest Neighbor classifier.** A classification system which calculates the distance between a sample point and all training samps; the new point is in the same category as the majority of its k nearest neighbors (p. 4).

**Osuna's method.** A method for training SVMs by fixing the size of the quadratic problem and cycling through the entire sample space (p. 5).

**QP – Quadratic Programming problem.** Memory and time intensive numerical method used to train SVMs using brute force and large matrices, enhanced by Osuna's method and SMO (p. 14).

**RBF – Radial Basis Function.** A nonlinear kernel used frequently in SVM training. The RBF kernel maps nonlinearly separable data into a different space so it may be used for classification (p. 71, see also p. 11+).

**SMO – Sequential Minimal Optimization.** A method for training SVMs by optimizing two vectors at a time analytically instead of using a numeric quadratic problem solver, developed by Platt in 1997 (p. 15).

**SV – Support Vector.** A feature vector selected through optimization for inclusion in a support vector machine model (p. 12).

**SVM – Support Vector Machine.** A system for performing automated classification of feature vectors by locating a hyperplane separating the target groups in multidimensional space, developed by Vapnik and Chervonenkis in 1979 (p. 5).

**UCE – Unsolicited Commercial E-mail.** Junk mail, also called spam (p. 55).

Chapter 1

# Introduction and Background

High-level textual and pictorial content recognition, classification, and sorting continue to be challenging and actively researched areas of interest. Programming a computer to perform binary classifications, such as identifying faces in an image or flagging news articles of interest to a particular person, represent a substantial theoretical and computational task. Support Vector Machines (SVMs), a type of feature-based classification system, provide an efficient and accurate mechanism for this type of automated classification [12]. SVM classification systems are capable of high-accuracy classification while maintaining acceptable model training time, sample testing time, and system storage requirements.

The purpose of this thesis is to examine the operation of the SVM sequential minimal optimization training algorithm and its application to two classification problems. The first section describes the process of training a SVM using sequential minimal optimization and two possible approaches to improve performance using parallel processing. The second and third sections describe the classification applications that are of particular interest for this thesis. The first problem, which deals with scanned consumer photographs, labels each photograph as depicting an indoor or an outdoor scene. The second problem, which uses textual features, labels electronic mail messages as junk or legitimate.

## 1.1 Feature-Based SVM Classification Systems

The construction of a feature-based classification system, such as one utilizing SVMs, occurs in two phases: training and testing (see Figure 1.1). In the training phase, a series of representative samples from both positive and negative categories are obtained and manually labeled. These samples are utilized by the training process to create a classification model. Training is generally a very lengthy, time-consuming process, and is performed only once. After a system model has been obtained, the system may be used to perform classifications of new samples very quickly.

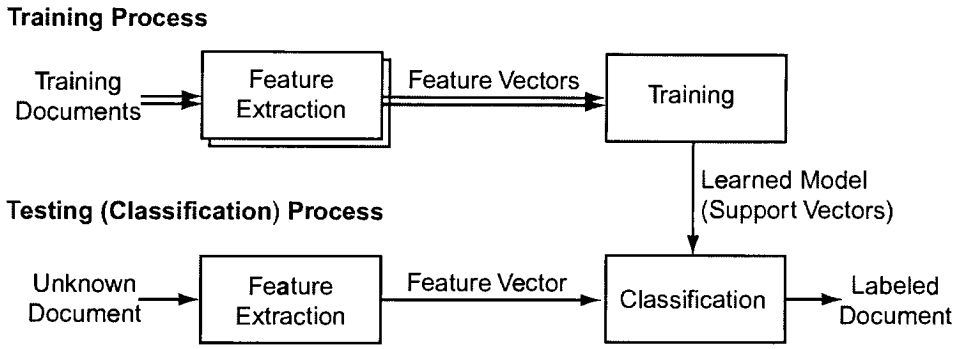



**Figure 1.1** – SVM data flow block diagram

SVMs do not classify complex raw documents, such as text or images, directly. Instead, the SVM training or classification process starts with a feature extraction stage. Feature extraction performs a consistent algorithmic reduction on a document, generating a simpler representative description of the sample's pertinent properties. The series of properties extracted from a sample is referred to as a feature vector, and the number of total features possible from all samples is the feature space. When working with text documents, for example, words can be features. The feature space contains all of the words in the extractor's dictionary, and feature vectors may count the number of times a

particular word appears in a document or whether or not a word appeared at all. Feature vectors generated by textual documents are generally sparse, and the feature vector contains a small number of identified features from a very large feature space. Feature vectors generated by examining images are typically dense, containing data from scaled metrics. For example, a color histogram may be generated by a feature extractor for a full image or blocked portions of an image, creating a feature vector with the number of features equal to the number of bins in the histogram. Regardless of the data type, the selection of an appropriate feature extraction scheme remains an essential component of effective classification system design. The selected feature vectors must adequately represent the difference between categories for classification to be possible.

## 1.2 Other Classification Techniques

SVMs are only one of several techniques available to perform automated classification based on feature vectors. Other techniques, such as inductive rule-based classifiers, neural networks, probabilistic and statistical models, and geometric nearest-neighbor classifiers, have also demonstrated usefulness in classifying a diverse range of data. The selection of a classification technique for an application requires evaluation of the method's training time, classification execution time, storage requirements and simplifying assumptions, as well as the ease of incorporating new documents into a previously trained system.

One of the oldest approaches to machine classification uses inductive learning to derive a set of rules given a set of samples [4]. Two well-known applications, called Construe and Ripper, use similar rule-based approaches. Operating directly on samples

instead of feature vectors, these algorithms produce a series of if-then clauses representing a logical conjunction which may be evaluated very quickly to classify a new document. Ripper, for example, constructs rules to designate which documents may be labeled in the positive class, and then includes assertions which may be used to prune false samples. Rule-based approaches have the advantage of being easily readable and modifiable by humans and allow users the opportunity to specify prior or posterior constraints. Unfortunately, the rule construction process generally requires an extreme simplification of a document's feature space, and can still require substantial computational time.

Neural networks and probabilistic classifiers are two other approaches to automated classification. Neural networks can be constructed to learn nonlinear mappings between features and categories [17], although acceptable performance is achieved only by limiting the dimensionality of the feature space. Probabilistic classifiers, such as the naïve Bayes classifier, assume that individual features are completely independent, and again require dimension reduction [9].

Another classification model, the k-Nearest Neighbor (k-NN) classifier, uses a geometric approach to determining sample similarity [9]. Each sample is mapped into a multidimensional feature space using a function which groups similar samples in clusters. To classify a new sample, the distance from the new point to every training sample point is calculated, and the new point is labeled the same as the majority of its $k$ nearest neighbors. The evaluation time of the classifier is directly related to the number of samples contained in the system, and can become prohibitively large if a complex nonlinear mapping function must be evaluated to calculate relative distances.

SVMs reduce the evaluation-time computational complexity present in k-NN classifiers by determining the separation boundary between the two cases. Instead of comparing a new point to every training point, a SVM simply determines if a new point is above or below the separating boundary to produce a classification. Training a SVM requires locating this separating hyperplane, and forms the basis of SVM theory development.

## 1.3 Support Vector Machines

The statistical learning theory behind SVMs was originally developed by Vapnik and Chervonenkis in 1979, and its related concepts of structural risk minimization and support vector regression are frequently referred to as VC theory [2][7]. The original mathematical foundation of SVM training requires the minimization of a function expressed as a Lagrangian represented by a matrix, so brute force training makes extensive use of a quadratic programming (QP) solver in its innermost loop. Several approaches to facilitate faster and less computationally intensive SVM training have been developed [11][12].

Training a single large SVM with QP methods uses substantial amounts of memory by requiring a matrix capable of storing the number of samples squared [12]. Vapnik initially suggested training a SVM with a technique known as chunking, which solves portions of the QP problem to identify and remove rows and columns evaluating to zero, thus reducing the size of the problem before solving it completely. To increase efficiency and further reduce memory requirements, Osuna [11] proposed a new procedure for SVM training. Osuna's method solves the complete QP problem as a series

of smaller QP subproblems. The algorithm maintains a constant size QP matrix in the solver, adding new untrained samples while removing properly trained samples between each step. Both methods, however, still utilize QP solvers, which Platt notes are nontrivial and require substantial attention to detail: "Numerical QP is notoriously tricky to get right; there are many numerical precision issues that need to be addressed" [12, p. 5]. Most numeric SVM solvers utilize professionally designed QP packages to perform the appropriate computations during the algorithm's execution.

To eliminate the complexity and inefficiency of using a QP subroutine in SVM training, Platt [12] proposed a new training technique called sequential minimal optimization (SMO). SMO uses Osuna's algorithm to decompose a SVM training problem into a series of the smallest possible QP problems, each consisting of exactly two vectors, which are then optimized analytically. The SMO algorithm uses simple iterative heuristics to select vectors for optimization, terminating the training process as soon as a valid solution has been obtained. On the most drastic of Platt's test cases, the SMO method reduced training time from over 5 hours to 17 seconds, while other representative problems were reduced from requiring multiple hours to only several minutes. SVMs, trained with SMO, provide a fast and accurate classification system.

Recently, SVMs have become a prominent classification system for both images and text for a variety of applications [2][8]. For the purposes of this paper, two classification systems will be examined. The first application attempts to label consumer photographs as having been taken indoors or outdoors, while the second classifies electronic mail messages as junk or legitimate.

## 1.4 Image Classification

Szummer [15] demonstrated a successful indoor/outdoor image classification system utilizing low level features, including color, texture, and frequency data. Each image was partitioned into 16 subblocks, consisting of 4 rows and 4 columns, and analyzed to produce color features and texture features using a multiresolution simultaneous autoregressive (MSAR) model. The entire image was analyzed to produce a frequency feature using the discrete Fourier transform and discrete cosine transform. Each block was classified independently, and the intermediate block-based results were presented to another classifier to produce a final classification.

Szummer used both k-NN classifiers and neural networks to perform the actual classification task, but remarked that training the neural network was excessively slow and produced results worse than the k-NN classifier for the color features. Two important conclusions are presented in the research. First, classifying subblocks and combining judgments produces better results than attempting to classify a full image at once. Second, combining the predictions of two independently weak features with an additional classifier produced better results than selecting a single good feature. Overall, Szummer's best results were obtained when using the color and MSAR texture features, combined with a secondary classifier, achieving an accuracy of 90.3%.

Serrano [14] decreased the complexity of the system by reducing the feature space and adopting more efficient feature extractors. By reducing the number of color histogram features from Szummer's 96 points to 48, and utilizing a more efficient wavelet algorithm to extract texture instead of the MSAR model, Serrano reduced feature extraction time from several hundred seconds to less than half a second. In addition,

Serrano adopted a SVM with a radial basis function (RBF) kernel to perform the classification. The two stage classification scheme was again utilized, and produced results with 90.2% accuracy in substantially less time, demonstrating an appropriate balance between feature set reduction and classifier performance.

## 1.5 Electronic Mail Classification

Joachims [8] demonstrated that SVMs are also an appropriate mechanism for categorizing textual documents based on content because of their sparse feature sets and large feature spaces. Joachims compared the accuracy and performance of SVMs to Naïve Bayes, k-NN, rule-based decision tree, and linear Rocchio classifiers by categorizing Reuters and medical journal articles into predefined categories and found that SVMs, independent of numeric system parameter selection, performed more accurately than the four other classification techniques. The SVM required excessive training time, but was substantially faster in evaluating new samples than the k-NN classifier. Kwok [9] independently produced similar results using a SVM with Osuna's algorithm.

Several feature extraction techniques are possible for textual documents. The simplest, which produces sparse binary feature vector, merely includes an entry if a particular dictionary word was present in a document. The classifier may be extended from signifying presence by noting the number of times a word appears in a document. Both Joachims [8] and Kwok [9] utilize a slightly more complex term frequency – inverse document frequency (TF IDF) method for feature extraction. In the IDF coding, a single word's IDF frequency is calculated using

$$IDF = \log\left(\frac{n}{DF}\right) \quad (1)$$

where $n$ is the number of training documents and the document frequency (DF) the number of documents the specified word appears in [8]. The TF IDF value for a document is then obtained by multiplying TF by IDF. Using this method, words which appear in only one document have high TF IDF values, while common words produce low TF IDF values.

Sahami [13] presented one of the first applications of probabilistic classification techniques to label electronic mail as unsolicited junk or legitimate personal and business communications. By using factor analysis to select 500 words as features for classification, Sahami reduced the feature set's dimensionality to a magnitude capable of reasonable use with a Bayes classifier. The system produced accuracy results between 87.7% and 97.1%, leading Sahami to conclude that probabilistic mail categorization is feasible but suggested applying Joachim's SVM research in future work.

Drucker and Vapnik [4] applied SVMs with SMO to categorize junk electronic mail. Drucker found that the most accurate results were produced using simple binary features, constructed without regard to case and without use of a manually generated word exclusion list, as opposed to the TF-IDF representation. Further, the best results were obtained by utilizing the entire feature space instead of an extracted subset.

SVMs have demonstrated successful results when applied to these image and electronic mail classification applications. This thesis hopes to decrease the amount of time required to train a support vector machine through parallelization, further increase performance in image classification through parallel feature extraction, and apply the SVM classifier to the electronic mail junk transmission detection problem. The content of

this thesis is organized into five remaining chapters. The second chapter examines the theory behind SVM training and evaluation, with an emphasis on the SMO training method. The third chapter presents a parallel implementation of the SMO algorithm. The fourth and fifth chapters develop the image scene categorization and electronic mail classification applications, while the sixth chapter concludes.

Chapter 2

# Support Vector Machine Theory and Operation

SVMs facilitate fast binary classification by constructing a separating hyperplane between positive and negative samples. SVM theory presents a methodology for identifying the location of the separating hyperplane in an arbitrary multidimensional system so that a new sample may be classified by merely determining whether the point is above or below the separating plane. In a linearly separable system, the samples are already clearly separated and the hyperplane may be easily located. In some systems, however, the positive and negative samples may not be separable with a single plane, and the SVM requires the use of an additional tolerance parameter to enable separation. Further, in nonlinear systems, a kernel mapping function may be used to transform raw feature data into a linearly separable system for use with SVMs. The process of locating the hyperplane, known as training, involves solving a complex QP problem directly or iteratively using SMO.

## 2.1 Support Vector Machines

In a SVM system, a number of samples must be considered to establish the model. These samples, whose classifications are known in advance, are designated as training samples. Each sample $i$ of N total samples is represented by its feature vector, $\mathbf{x}_i$, and its already known classification, $y_i$, which may be +1 for a positive example or –1 for a negative example. The goal of the SVM is to identify a hyperplane which separates the

samples of the two classifications by as large a margin as possible (see Figure 2.1). The samples closest to the separating hyperplane, shaded in the figure, are the model's SVs. In a linearly separable system, the SVM may be represented by a single weight vector **w**, which contains one element for each dimension of the feature space [12]. Each sample is required to fall on the appropriate side of the of the hyperplane, such that

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \text{ for } y_i = +1 \qquad (2)$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \text{ for } y_i = -1$$

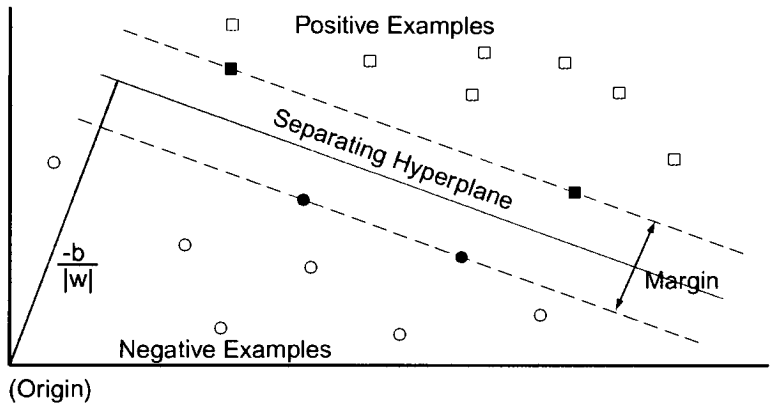



Figure 2.1 – Separating hyperplanes in a linear SVM [2]

Once the separating hyperplane has been identified, any given sample with feature vector **x** may be classified using the weight vector **w** by the equation

$$u = \mathbf{w} \cdot \mathbf{x} - b \qquad (3)$$

where $b$ is the SVM's bias parameter. In the final SVM, points along u = 0 lie on the separating hyperplane and belong to neither classification. This separating hyperplane lies directly between the closest points from the two categories, separated from the samples by an equal margin on both sides. Training samples lying on these lines, expressed where u = –1 or u = +1, are considered the support vectors of the system. The difference between these classifications is represented by margin

$$m = \frac{1}{\|w\|^2} \tag{4}$$

whose minimization serves as the basis for the computational optimization problem.

SVMs numerically solve this optimization by using a Lagrangian to convert the system into a dual form convex QP problem. The final objective function is dependent on a Lagrange multiplier for each training sample

$$L_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{5}$$

subject to the constraints

$$\begin{aligned} &\alpha_i \geq 0, \forall_i \\ &\sum_{i=1}^{N} y_i \alpha_i = 0 \end{aligned} \tag{6}$$

in the linear case [2][12]. In some circumstances, however, the system may not be separable with a hyperplane when requiring all of the training samples to be outside the margin. In this case, introduction of a tolerance parameter C allows but penalizes samples within the margin, changing the first constraint of the Lagrange multipliers to

$$0 \leq \alpha_i \leq C, \forall i. \tag{7}$$

SVMs may also be used to perform classification in nonlinear systems through the use of a kernel function. Classifying a nonlinear system requires substantially more computations than a linear system. In the linear SVM, the output could be determined with a single dot-product between the weight vector **w** and the new sample **x**. In a nonlinear SVM, determining a classification requires performing a dot product with the new sample **x** and every support vector $\mathbf{x}_i$:

$$u = \sum_{j=1}^{N} y_j \alpha_j K(\mathbf{x}_j, \mathbf{x}) - b \tag{8}$$

The final optimization problem then becomes [2][12]

$$L_D \equiv \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

$$0 \le \alpha_i \le C, \forall i$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0$$

For both linear and nonlinear SVMs, the optimum solution to the minimization problem may be recognized by applying the Karush-Kuhn-Tucker (KKT) conditions to every Lagrange multiplier [12]. For the system to represent a valid solution, each $\alpha_i$ must meet the following KKT conditions:

| | | |
|---|---|---|
| $\alpha_i = 0$ | when | $y_i u_i \ge 1$ |
| $0 \le \alpha_i \le C$ | when | $y_i u_i = 1$ |
| $\alpha_i = C$ | when | $y_i u_i \le 1$ |

A support vector is considered bounded when its $\alpha = C$ within a tolerance, and the remaining vectors with $\alpha$ values between 0 and C are considered non-bounded. Vectors with multipliers at $\alpha = 0$ are discarded as support vectors and removed from further consideration.

The process of solving a SVM produces Lagrange multipliers $\alpha_i$ for each training sample which satisfy the KKT conditions. Given these multipliers, a linear SVM's final weight vector may be calculated using

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \tag{10}$$

recalling that **w** is a vector containing an element for each dimension in the feature space. The bias parameter $b$ for any given sample may be determined using the evaluation function in reverse

$$b_i = \mathbf{w} \cdot \mathbf{x} - y_i \tag{11}$$

with the final bias for the entire SVM calculated as the average $b$ of all bounded support vectors. For nonlinear systems, the weight vector may not be used, so the bias for any sample may be determined directly using

$$b_i = \sum_{j=1}^{N} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) - y_i \tag{12}$$

with the SVM's final bias again calculated as the average of all bounded support vectors.

## 2.2 Sequential Minimal Optimization

Training a SVM using traditional matrix operations requires the use of a numeric QP problem solver, a complex time-consuming software subroutine. Osuna's algorithm reduces this computational burden slightly through a technique known as chunking. By training smaller portions of the entire SVM sample set in groups, vectors which are not SVs are discarded early, so only the potential SVs are trained in the final phase. Osuna's algorithm still requires the use of a QP solver. Platt's SMO algorithm [12] extends chunking by optimizing with the smallest possible subset at each possible step—two sample vectors—and eliminates the requirement for a QP subroutine by producing the optimization analytically. The SMO algorithm consists of two distinct components. The first component, implemented using nested loops, uses simple heuristics and the KKT conditions to select two $\alpha_i$ values to optimize. The second component of the algorithm

optimizes these two Lagrange multipliers, updating the overall system state and bias after each iteration.

The outer loop of the SMO algorithm ensures that every sample vector in the training set meets the KKT conditions before the optimization process is allowed to terminate. This loop alternates between checking all multipliers and checking only those which have not been bound between 0 and the margin tolerance parameter C. When this loop can complete an entire iteration through the sample set without detecting a KKT violation, the SVM is trained and the algorithm is complete.

When a Lagrange multiplier violating the KKT conditions is identified, the SMO algorithm attempts to optimize the offending sample. SMO optimizes multipliers in groups of two, so a second multiplier must be selected by the algorithm's inner loop to perform this joint operation. The inner loop considers all samples in the training set, prioritizing its selection in an attempt to increase the algorithm's efficiency. First, the algorithm considers samples with non-bounded multipliers ($0 < \alpha < C$) and selects the sample with the largest absolute error. If this selection cannot make positive progress, then the algorithm attempts to select any non-bounded sample. Finally, if these selections cannot progress, then the algorithm randomly selects a starting location in the training set and iterates through all samples until one is found which can perform the optimization.

The most substantial component of SMO is Platt's analytic solution for the optimization of two sample vectors. Because the entire system must satisfy the constraints

$$0 \le \alpha_i \le C, \forall i \tag{13}$$

$$\sum_{i=1}^{N} y_i \alpha_i = 0$$

the two-sample subset examined in each joint optimization step must also satisfy these constraints. The first inequality constraint limits the possible values for $\alpha_1$ and $\alpha_2$ to be inside a box. The linear equality constraint, represented in the summation among all targets and Lagrange multipliers, requires the selected multipliers to lie along a diagonal line (see Figure 2.2).

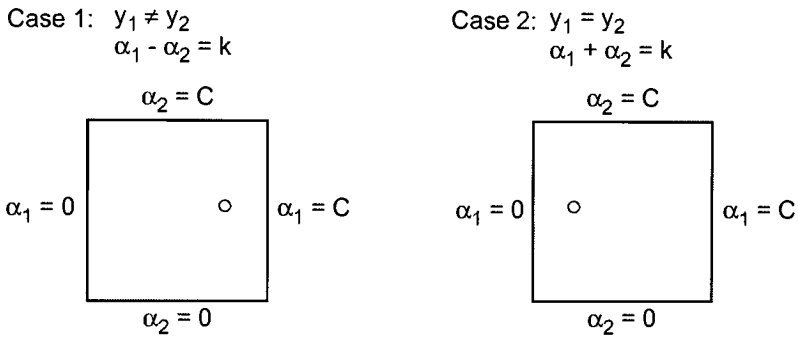



Figure 2.2 – Constraints for optimizing two Lagrange multipliers [12, p. 6]

The SMO algorithm first calculates $\alpha_2$ based on the ends of the diagonal line segment. If the samples being optimized are from different categories, then $\alpha_2$ is bound by

$$L = \max(0, \alpha_2 - \alpha_1) \tag{14}$$

$$H = \min(C, C + \alpha_2 - \alpha_1)$$

If the training samples are from the same category, then diagonal line is constrained by

$$L = \max(0, \alpha_2 + \alpha_1 - C) \tag{15}$$

$$H = \min(C, \alpha_2 + \alpha_1) .$$

Platt [12] showed that the second derivative of the objective function is

$$\eta = K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2) \tag{16}$$

which should, in normal conditions, be positive. In this case, SMO uses the second derivative with the error of each sample ($E_i = u_i - y_i$ ) to find the minimum point of the objective function along the diagonal line and calculate the new value for $\alpha_2$.

$$\alpha_2^{new} = \alpha_2 + \frac{y_2(E_1 - E_2)}{\eta} \tag{17}$$

This calculated value is then clipped by the ends of the line segment using

$$\alpha_2^{new} = \begin{cases} H \ \ if \ \ \alpha_n^{new} \geq H \\ \alpha_2^{new} \ if \ L < \alpha_n^{new} < H \\ L \ \ if \ \ \alpha_2^{new} \leq L \end{cases} \tag{18}$$

The clipped new value for $\alpha_2$ is used to determine the new value for $\alpha_1$:

$$s = y_1 \, y_2 \tag{19}$$

$$\alpha_1^{new} = \alpha_1 + s(\alpha_2 - \alpha_2^{new}) \tag{20}$$

In the event that n is not positive, the function is evaluated at the ends of the line segment. The final computational steps of the SMO algorithm perform necessary housekeeping functions essential to the efficiency of the algorithm. Considering the change in the Lagrange multipliers

$$\begin{aligned} t_1 &= y_1(\alpha_1^{new} - \alpha_1) \\ t_2 &= y_2(\alpha_2^{new} - \alpha_2) \end{aligned} \tag{21}$$

the system's potential new bias values are first calculated incrementally [12]:

$$\begin{aligned} b_1 &= E_1 + t_1 K(\mathbf{x}_1, \mathbf{x}_1) + t_2 K(\mathbf{x}_1, \mathbf{x}_2) + b \\ b_2 &= E_2 + t_1 K(\mathbf{x}_1, \mathbf{x}_2) + t_2 K(\mathbf{x}_2, \mathbf{x}_2) + b \end{aligned} \tag{22}$$

The system's overall bias b is set to $b_1$ or $b_2$ depending on whether or not $\alpha_1$ or $\alpha_2$ is at bounds. If $\alpha_1$ is not at the bounds, then the bias value $b_1$ is correct, as it forces the SVM to output a perfectly correct answer (target disposition $y_1$) given sample $\mathbf{x}_1$. If $\alpha_2$ is not at

the bounds, then its bias value is similarly correct. If both $b_1$ and $b_2$ are valid, then the SMO algorithm chooses the average of the two choices.

After updating the bias, the error caches $E_1$ and $E_2$ are set to zero because the just-optimized pair of samples contain no error with respect to the new value for the bias. The error cache values of every other sample, however, must be increased by the change in the bias. This change may cause previously optimized samples to violate the KKT conditions, requiring that they be optimized again. The SMO algorithm's dual-loop iteration then continues selecting samples for joint optimization until all samples meet the KKT conditions, indicating a valid SVM system.

Chapter 3

# Parallelization of Sequential Minimal Optimization

Training a large SVM with the SMO algorithm still requires substantial amounts of time. Due to its simple iterative nature and sizable data set, SMO is a prime candidate for execution time reduction through parallelization. Despite its straightforward operating characteristics, SMO contains several data dependency details which require consideration and restrict its capacity for parallel operation. This paper presents two distinct approaches for parallelization. The first approach, a distributed execution implementation, places the entire SVM on all of the processors while specifying ranges for parallel optimizations. This maintains SMO's capability of optimizing any vector pair within each iteration. The second approach, a blocked independent parallelization, trains a single SVM as several completely separate SVMs and combines the results. Both approaches present mixed results considering execution time, final satisfaction of the KKT conditions, test set evaluation accuracy, and termination constraints.

The introduction of the SMO algorithm substantially reduced SVM training time from previous QP-based solution methods, but extremely large data sets can still require lengthy amounts of time to produce a model. For example, training the Adult-7 census data subset containing 16,100 samples with a dimensionality of 120 features on a HP Visualize workstation required 13 minutes with a linear kernel but 31 hours with a RBF kernel. In addition, SMO's nested loop optimization selection heuristic has an execution time of $O(n^2)$ based on the number of samples, so increasing the sample set size

dramatically increases the training time. This dismal solution time suggests that SVM is a candidate for parallelization.

A cursory examination of the SMO algorithm [12] and two previous implementations [1][6] indicates that SVM has parallelization potential with an independent optimization step beneath an iterative element. At its lowest level, SMO repeatedly executes a segment of code that optimizes two Lagrange multipliers. This optimization step is self-contained and capable of optimizing any two given multipliers when called from the innermost of two nested loops. The outer loop simply iterates across all of the samples sequentially, verifying that a sample meets the KKT conditions. When a violation is found, the inner loop selects another sample for joint optimization. An optimization modifies exactly two Lagrange multipliers and adjusts a new SVM bias value. Thus, SMO only optimizes two Lagrange multipliers at any given time, and does so in a predictable order.

The iterative nature of the SMO algorithm suggests a simple parallelization approach. In the sequential implementation, one processor iterates across the entire set of samples. In a parallel system, the set of samples may be partitioned into blocks. Each block is assigned to a processor, which runs the SMO algorithm on its assigned samples. However, the SMO solution is not perfectly independent at the block level. Arbitrarily partitioning a SVM into blocks and training on multiple machines simply produces multiple independent SVM solutions. Several data dependencies within the SMO optimization process require detailed consideration for an appropriate parallelization strategy.

The first aspect of the SMO algorithm requiring attention before parallelization is its utilization of a global bias parameter. The bias parameter is essential to the proper execution of the SVM, as this single floating point value is used as a linear offset when executing the evaluation function to classify a new sample. The bias is used in the evaluation equations

$$u = \mathbf{w} \cdot \mathbf{x} - b \tag{23}$$

in a linear SVM or

$$u = \sum_{j=1}^{N} y_j \alpha_j K(\mathbf{x}_j, \mathbf{x}) - b \tag{24}$$

in a SVM with a nonlinear (e.g. RBF) kernel. Clearly, this single value is of fundamental importance to the accuracy of any SVM. Irrespective of its importance, calculating the bias parameter is ambiguously defined in the literature. Platt's SMO algorithm [12] calculates the bias parameter incrementally by using a complex expression to adjust the bias in the positive or negative direction based on the change in the Lagrange multipliers following each iteration. Joachims [8, p.6] suggests selecting two arbitrary support vectors, one from the positive class and one from the negative class, finding $b$ for these samples, and using their average as the system's bias. Burges [2, p. 11] notes that it is "numerically safer" to calculate the bias value by using the arithmetic mean with $b$ values from all samples with $\alpha_i \neq 0$.

The single bias value is partially responsible for the large number of iterations required for a sequential SVM to converge to a solution when using SMO. For a solution to be valid, each support vector must meet the KKT conditions and be classified properly. The classification for each sample is determined by evaluating its output using the current bias and comparing it to the target value. When an optimization step completes, the bias

changes. Thus, a successful optimization step may adjust the bias and cause previously optimized samples to violate the termination conditions. As subsequent optimization steps complete and the bias changes multiple times, the initially optimized samples may no longer be valid. To correct for this situation, the SMO outer loop always performs an additional complete iteration through the data set to check for new KKT violators before terminating the algorithm.

In a parallel implementation of the SMO algorithm, the single bias value becomes even more problematic. As multiple processors train their subblocks independently, each optimization step produces a new local bias value. When the iteration completes, each processor has a local bias value. Before the next iteration may begin, the bias values must be synchronized across processors. This step causes a portion of each processor's samples to become KKT violators, requiring additional training iterations and slowing convergence if convergence is still possible.

The second consideration for parallelization is SMO's flexibility in selecting samples for joint optimization. The inner loop of SMO may use three heuristics to identify a vector. The algorithm's first choice is a non-bounded SV, which generally ensures positive progress. If no non-bounded SVs are available, then the algorithm prefers any previously optimized vector and then any sample at all. Based on this heuristic, SMO's training preferences are predictable. A parallel implementation utilizing any type of blocking introduces artificial restrictions which may decrease the availability of samples for joint optimization. For example, SMO may use the first choice heuristic to optimize a sample and a SV near the end of the data set. In a parallel implementation, the processor may not have access to that SV, forcing it to select a less optimum sample with

the second or third choice heuristic. Ensuring the availability of all-to-all optimization may be a priority for an efficient and accurate parallel SMO SVM.

Most SMO implementations also utilize an error cache and a linear weight vector to enhance the performance of the algorithm. These techniques are not conducive to a parallel approach, and require extra computational steps to ensure a correct solution. SMO spends most of its time evaluating the output of a sample vector based on the current state of the SVM. Comparing the current output to the desired target output produces the error for a sample, which is used with the KKT conditions to determine termination. Evaluating the output of a vector can be costly in itself, but requiring large numbers of these evaluations presents a substantial computational requirement. To avoid this bottleneck, SMO uses an error cache for all samples. At the end of a joint optimization step, SMO calculates the differential error that the change in bias value introduces for all other samples. The recently optimized sample has no error, but all other samples accumulate a very small error based on the evaluation of the kernel function. This change in error must be processed for every other sample in the entire system to maintain a coherent error cache.

In any type of parallel approach, maintaining a consistent error cache for all samples across all processors is not possible. Each sample's new error is based on the current system bias, changes after every optimization, and now differs across processors. Therefore, the error cache may be maintained only within the context of a local processor's optimization progress. After a communication step, where results from other processors are integrated, the error cache must be cleared and reinitialized using the new shared bias value.

Another technique used to increase the efficiency of the SMO algorithm is a single linear weight vector. In a linear SVM, the output of any sample vector may be evaluated by a single dot product with the weight vector to obtain a classification. This is much faster than summing a series of dot products with each support vector. In a parallel implementation, different processors generate different weight vectors, and the weights must be recalculated after a communication step. The weight vectors may be easily generated by examining the support vectors and their corresponding Lagrange multipliers.

To address the various data dependencies of SMO's algorithm, two different parallel systems were implemented. The interleaved method places the entire data set on all of the processors and specifies which subblocks of the entire problem may be optimized by a processor at which time. The blocked system uses a simpler approach, subdividing a large data set into smaller blocks on each processor, trains independently, and roughly attempts to combine the results.

## 3.1 Interleaved Parallelization

The interleaved parallelization attempts to maintain all of the constraints of uniprocessor SVM training while distributing the task of vector optimization to multiple processors. The entire sample set is loaded onto all $n$ processors participating in the training process and logically subdivided into $2n$ segments, so that each processor contains two segments of sample vectors. To ensure the possibility that any support vector has the opportunity to optimize with all other support vectors, an all-to-all optimization pattern is maintained (see Figure 3.1 and Figure 3.2).

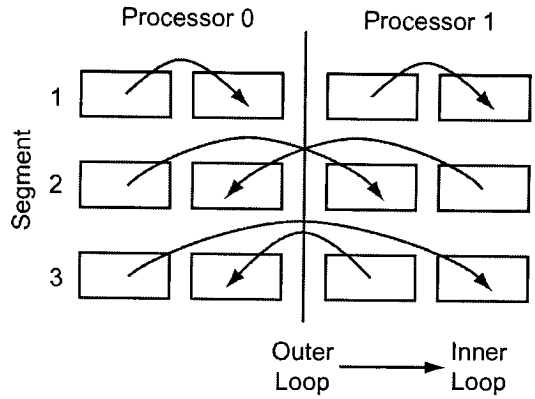



Figure 3.1 – Optimization pattern for two way parallel interleaved approach

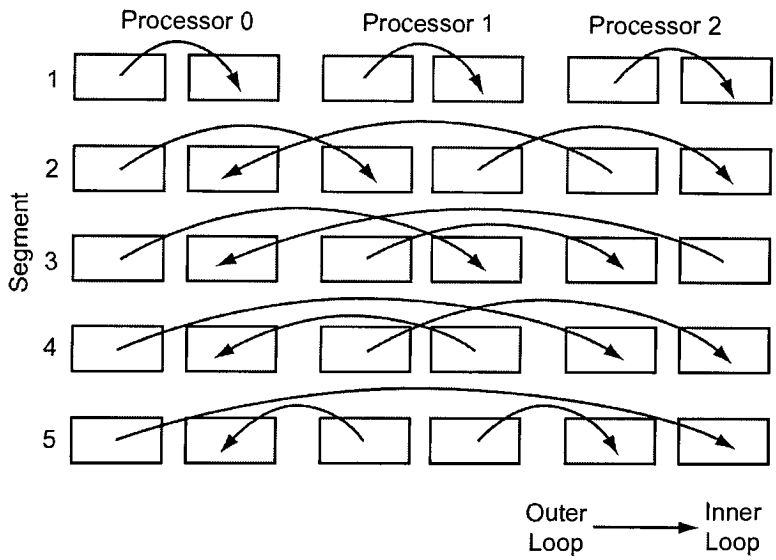



Figure 3.2 – Optimization pattern for three way parallel interleaved approach

The optimization pattern specifies which segments on each processor are permitted to optimize at a given stage during the system's operation. As all of the segments are on all of the processors, this selection is arbitrary but sequenced specifically to prevent two processors from optimizing the same segment at the same time. When the

optimization sequence starts, the bias, Lagrange multipliers, and error cache are identical across all of the processors. Each processor begins to run the SMO algorithm on its assigned segments, looping until both converge to a solution. During these optimization steps, the Lagrange multipliers and error cache are updated only for the segments under consideration.

After each processor's set has converged to a local solution, the results are disseminated. Each processor sends its new Lagrange multipliers to every other processor, so the results of the optimizations are available to the entire system. The local bias values are averaged using a MPI allgather operation, which computes the arithmetic mean of all local bias values to produce a global bias for the system's subsequent calculations. The optimization process continues so that each segment is jointly optimized with every other segment. In a 2 processor system, this requires 3 optimization steps, while a 3 processor system requires 5 steps. After all possible segment optimization permutations have been performed, the system is examined for termination conditions. To determine if the system has completely converged to an appropriate solution, the number of changed multipliers during each step is summed for the entire system. If no multipliers have changed, then the algorithm terminates. If any multipliers changed, then the communication sequence begins again.

## 3.2 Blocked Independent Parallelization

While providing a general all-to-all optimization pattern, the interleaved parallelization approach has the immediate disadvantage of complexity, substantial communication and computation requirements, and poor scalability. As the system's

parallelization increases, the number of communication exchanges increases, and each exchange requires the immediate time-consuming recalculation of the SMO error cache. These factors indicate that the interleaved parallelization may be counterproductive, as increasing parallelization dramatically increases the computational workload required to continue the algorithm's basic operation.

In order to achieve acceptable performance increases, an additional amount of independence must be introduced into the parallel SVM system. Etin [5] suggested that a completely independent parallelization may be performed with only a small loss in accuracy by adjusting the method used to compute the new bias value after local optimization. Instead of using the simple arithmetic mean between all processors, Etin proposed using a weighted average of all SVs across all processors. Each processor multiplies its local bias value by the number of SV on that processor, and the result is summed across all processors. Dividing by the total number of SVs produces the new global bias value.

The blocked independent parallelization divides the samples equally among the $n$ processors participating in the training process. Each processor trains completely independently. After each converges to a local solution, the resulting SVs are concatenated to create the final SV collection for the entire system and the final bias is computed using the weighted mean. This produces a complete, usable SVM, although some samples may violate the KKT conditions due to the change in bias from the local value to the system average. These violations are simply ignored, slightly decreasing the system's accuracy but substantially increasing performance.

## 3.3 Results

The SMO algorithm and pseudocode provided by Platt [12] as well as the implementation notes and code references from deAlmeida [1] and Ge [6] were used as the basis for constructing a modular sequential SVM software application in C++ (see Appendix A). To ensure the accuracy of the system, the results from this SVM were compared to Joachims' svmLight [8] as well as the implementations provided by deAlmeida and Ge. Training sets used for comparison included several standard data sets, including the UCI Adult data set with both linear and RBF kernels, the tic-tac-toe artificial data set, the Pima diabetes data set, and an arbitrary small test case containing meaningless yet separable points. In all cases, the output of the custom sequential SVM was consistent with the three published SVMs.

The interleaved parallel implementation provided deceptively positive results when trained with the UCI Adult-2 data subset with a linear kernel. The system converged quickly, producing a solution identical to the sequential case in only slightly more time. Tests with the RBF kernel, or other subsets of UCI-Adult, failed to converge entirely. Sets which converged quickly with the sequential SVM could train for several iterations on the parallel SVM becoming very close to a solution without meeting the conditions required to terminate as one or two samples would change every iteration. In some cases, this small change pattern would continue indefinitely. In other cases, the bias value on one processor would begin to accumulate in a particular direction, causing the average bias for the entire system to follow. As the system bias value diverged dramatically, more and more samples would violate the termination conditions, until

these small changes accumulated and caused the bias to explode by several orders of magnitude rendering the training process with the interleaved approach unsuccessful.

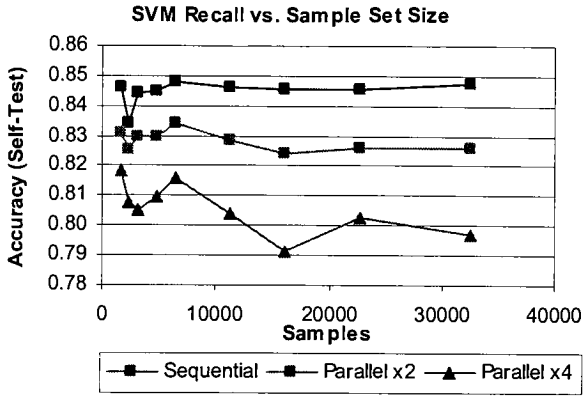



Figure 3.3 – SVM recall for Adult Linear

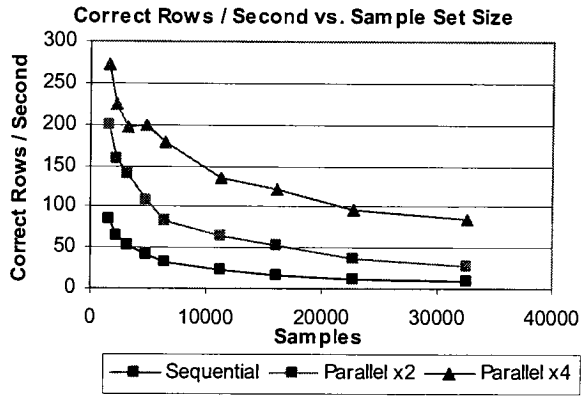



Figure 3.4 – SVM performance for Adult Linear

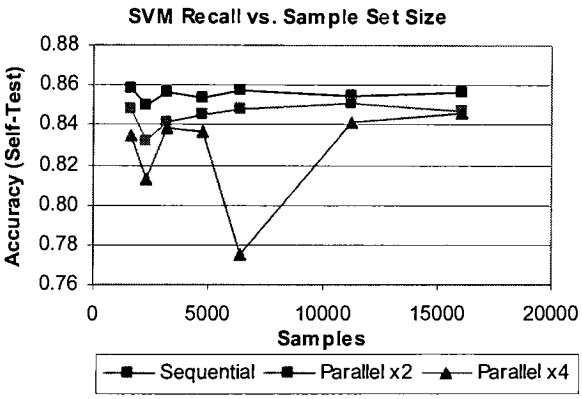



Figure 3.5 – SVM recall for Adult RBF

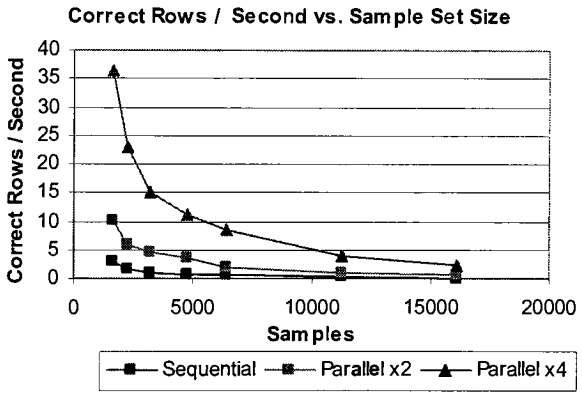



Figure 3.6 – SVM performance for Adult RBF

The blocked independent parallel approach substantially improved training time while decreasing recall accuracy only slightly (see Appendix B). In the case of the Adult data set with the linear kernel, accuracy was reduced from about 85% to 83% for the 2-processor parallelization and 80% with four processors (see Figure 3.3). The same data set with the RBF kernel demonstrated only a 1% loss in accuracy from 86% to 85% for the 2-processor system, while the 4-processor implementation varied in accuracy from 77% to 84% (see Figure 3.5). Accuracy is reported in terms of the SVM's recall rate,

which is produced by testing the SVM with the same data used for training. In an ideal system the recall rate would be 100%, as the classifier had previously examined every testing sample during the training process. Recall is used here as a simple metric to determine the relative change in the SVM's accuracy due to the parallelization process. Subsequent tests using the Adult data set demonstrated that the system's accuracy rate was maintained when generalized to larger, previously unexamined data sets.

When comparing parallel performance, the additional metric of correct rows per second is particularly useful. In the sequential SVM with the Adult data set and the linear kernel, the training process produces 19 to 84 correct rows/second. Two processors in parallel, however, produced 28 to 199 correct rows/second, while four processors achieve 84 to 272 correct rows/second (see Figure 3.4). Training the Adult-7 data subset with the RBF kernel with the sequential SVM produced only 0.12 correct rows/second and required 31.01 hours to complete. The four processor implementation produced 2.35 correct rows/second, requiring only 1.61 hours speeding up the process by a factor of 19.26 (see Figure 3.6). In all cases, SMO's correct row per second output rate is dependent on the problem size. Training a small set instead of a large set not only takes less real time, but also processes vectors at a faster rate. Even though training in parallel slightly decreases the system's accuracy, it produces substantial performance improvements.

## 3.4 Discussion

Training the SVM in parallel using SMO generally produces multiple independent SVMs instead of one consistent, distributed SVM. The most important consideration then

concerns the consolidation of the independently trained sets into one single SVM model. The interleaved approach performs this consolidation as frequently as possible, attempting to maintain a valid SVM across all processors at all times through frequent communication. The blocked approach lets the processors train independently and then attempts to merge the results into a valid model solution.

Both approaches to parallel training possess one substantial drawback which limits the validity of the solution, speed of convergence, and even the possibility of convergence: the system's operation depends on the initial distribution of the data. Consider the extreme case of a sample set with several thousand negative samples and several dozen positive samples. If all positive samples are located on one processor, then only that processor will be able to produce a valid SVM solution while the others will fail to converge because no separating hyperplane may be identified. Less dramatic situations produce systems with bias values that are wildly divergent across processors. This destructive interference skews the average when calculating the overall system bias. The interleaved parallelization does not handle these inconsistencies gracefully, as small repetitive changes prevent termination and may accumulate destroying the validity of the solution. The blocked parallelization, on the other hand, is unaware of such inconsistencies until the training has already terminated.

The loss in accuracy during blocked parallel training is related to the final averaging of the bias values. After each processor finishes training, the independent bias values are averaged using a weighted mean to produce the final system bias. This process may change the disposition of some previously trained samples and cause KKT violations. In the other implementations, these new KKT violators prevented termination

as another iteration was required to optimize the offending samples. The blocked implementation allows these final violations which contribute to the decrease in accuracy.

As the experimental results show, allowing a small number of KKT violations in the final SVM does not invalidate the entire system but produces only small percentage drops in accuracy. This is due to the small number of samples bumped into KKT violation by the bias averaging process. Specifically, the only samples invalidated are those whose outcomes are directly dependent on a specific bias value. When the change from the local bias to the global bias is greater than the sample's output disposition, the sample becomes a violator. Etin [5] graphically demonstrated that these are the SVs which are the closest to the separating hyperplane; that is, the samples which were most ambivalent in the classification stage even before parallelization.

Overall, parallelization using the blocked method provides speedup gains not only by using multiple processors but by reducing the number of samples each processor must train. The training time for a SMO SVM is $O(n^2)$, so partitioning the sample into multiple independent blocks reduces the training time for each block even before parallel processing is introduced. While final production applications will generally require the additional percentage of accuracy lost with the parallel SVM, the independent parallelization can provide similar generalized results in less time.

Chapter 4

# Parallel Image Classification

SVMs have demonstrated usefulness in classifying consumer photographs as depicting either an indoor or outdoor scene, which is a useful component of image indexing and retrieval applications [14]. Image classification typically uses low-level features, such as histograms with a variety of filters and color spaces, to extrapolate high-level scene properties, such as location. Some of the most influential work in this area, performed by Szummer [15] in 1999, demonstrated that such categorization is feasible and can be adequately accurate, while subsequent work by Serrano [14] introduced SVMs for two-stage classification and significantly simpler feature extractors reducing the solution's computational complexity. This paper exploits the image classification system's inherent component-level and data-level parallelism to further reduce computational time by using a cluster of ordinary workstations with the MPI parallel programming environment. In addition, the previous SVM product is used to produce an integrated image classification system, capable of producing scene judgments directly from image files without the data management discontinuities generally present in experimental classifier research scenarios.

The indoor/outdoor image classification system was constructed using an incremental process. Three separate low-level feature extractors were selected with the consideration of preliminary tests using a prototype SVM. Each feature extractor was then tied directly to a trained SVM and used to produce intermediate results. These

results were then combined logically, arithmetically, and using a second-stage SVM to produce a final judgment for each image in the test suite.

For testing purposes, this experiment utilized the Kodak consumer image database also used by Szummer [15], Serrano [14], and Muller [10]. The database consists of 1304 images, scaled and rotated to 384x256 pixels, containing 24-bit color subject to the normalization and equalization process used in the previous research. Two testing suites were used to provide numeric validation of results. The first suite, referred to as the 2-way test, divided the images into two sets of 652 with the number of indoor and outdoor images balanced between the two sets. One set was used for training and the other for testing, and the analysis was then reversed. The second suite, referred to as the 4-way test, divided the images into four sets of 326. One set was used for training, and the other three sets were used for testing. The training process was then repeated for each set using the others for testing.

Feature set selection for image classification generally encompasses color features and texture features. The simplest color feature is a color histogram (CH), which generally presents the least performance with the least computational requirement. Texture features may be generated using a variety of methods, such as the MSAR model used by Szummer [15] or the wavelet approach suggested by Serrano [14]. Keeping the complexity as minimal as possible, this system uses two color features, CH and color coherence (CC), as well as a edge histogram (EH) texture feature, in conjunction with a two-stage SVM classifier to produce classification results.

## 4.1 Color Feature Extraction

The CH is the simplest feature used in the image classification application. To generate a color histogram, the image's three color channels—red, green, and blue—are processed separately. The histogram is essentially a summation which counts the number of pixels in an image set to a particular color. When used as a feature vector, histograms provide freedom in the selection of bin size and color space.

For this application, 8-bin and 16-bin histograms are considered. The histogram for a channel is generated by iterating through all of the pixels in the image and dividing the color value, always between 0 and 255, by the number of bins, which produces the number of the bin containing the color. The corresponding histogram bin is then incremented. The histograms for the R, G, and B channels are computed separately and concatenated to form the final feature vector for classification. Thus, the 8-bin extractor produces a 24-element feature vector and the 16-bin extractor produces a 48-element feature vector, which are then presented to the SVM for classification.

The use of a simple RGB histogram was discouraged by Szummer [15], who noted that the results are "only somewhat better than just guessing that each image is outdoor" [19, p. 2]. More accurate results are produced by translating the image into the Ohta color space using the transformations

$$L = (R + G + B) / 3 \tag{25}$$

$$s = (R - B) / 2 + 128$$

$$t = (R - 2G + B) / 4 + 128$$

These specific transformation equations are similar, but not identical, to those used by Szummer [15] and Serrano [14]. The scaling and shifting components, produced by the

division and addition with constants, is included to ensure that all 8-bit RGB pixels translate into 8 bits for each of the Lst channels.

To aid in the selection CH extractor parameters for the classification system, a preliminary analysis was performed using 8 and 16-bin CH extractors with both the RGB and Lst color spaces (see Table 4.1 and Appendix C). The feature sets were trained using a SVM with C=10 and a RBF kernel where $\sigma = 1$ using the 4-way testing methodology. As anticipated, Lst features were more accurate than the RGB features, but only by a single digit percentage. The 16-bin features were also slightly more accurate than the 8-bin features for both color spaces.

| Feature | Resolution | Color Space | Accuracy |
|---|---|---|---|
| Color Histogram | 16-bin | RGB | 71.01% |
| Color Histogram | 8-bin | RGB | 71.37% |
| Color Histogram | 16-bin | Lst | 72.03% |
| Color Histogram | 8-bin | Lst | 71.50% |

Table 4.1 – Full image individual CH classifier results for 4-way test

For the final image classification system, the 8-bin Lst CH extractor was selected as it provides the second-best accuracy with the fewest number of features. Although the feature extraction times for the 8-bin and 16-bin histograms are identical, the SVM training and evaluation time greatly favors vectors of smaller dimensionality. To improve the SVM's efficiency and produce results consistent with Muller's hardware-based feature extraction approach [10], only 8-bin/channel feature vectors are included in the system.

In addition to CH, another color-based feature, color coherence (CC), was selected for inclusion in the classification system. As suggested by Muller [10], CC is a

filtered histogram which presents information regarding the similarity of colors constituting an image. Each pixel in the image is examined once, and if the color values of all the pixels on a masked region surrounding the pixel under consideration are within a threshold value, then the corresponding histogram bin is incremented. For this CC feature extractor, the color value is converted to grayscale using the Euclidean distance among the R, G, and B channels with a 4x4 pixel mask and compared to a fixed threshold of 24.

Preliminary tests indicated that CC is a particularly effective feature extractor for indoor/outdoor classification (see Table 4.2 and Appendix C) with results substantially exceeding those obtained by CH alone. Surprisingly, the RGB CC feature is more accurate than the Lst CC feature. The 8-bin RGB CC feature extractor was therefore selected for inclusion in the image classification system.

| Feature | Resolution | Color Space | Accuracy |
|---|---|---|---|
| Color Coherence | 8-bin | RGB | 79.65% |
| Color Coherence | 8-bin | Lst | 78.76% |

Table 4.2 – Full image individual CC classifier results for 4-way test

## 4.2 Texture Feature Extraction

Previous classification systems have generally included a texture-based feature, using either the MSAR model or a wavelet approach. MSAR is computationally expensive and wavelets numerically complex, so the edge histogram (EH) approach suggested by Muller [10] greatly simplifies the system. EH is calculated by performing a

standard color histogram after the application of the Sobel edge detection operator on each channel in the image.

| | | |
|---|---|---|
| $x_0$ | $x_1$ | $x_2$ |
| $x_3$ | $x_4$ | $x_5$ |
| $x_6$ | $x_7$ | $x_8$ |

Figure 4.1 – Pixel enumeration for 3x3 grid Sobel operator

The pixels for the Sobel operator mask are defined in the usual increasing row-major fashion (see Figure 4.1). The operator itself is a weighted mask which produces first-order derivatives in the $x$ and $y$ directions through the difference equations [16]:

$$\begin{aligned}\frac{\partial f}{\partial y} &\approx (x_6 + 2x_7 + x_8) - (x_0 + 2x_1 + x_2) \\ \frac{\partial f}{\partial x} &\approx (x_2 + 2x_5 + x_8) - (x_0 + 2x_3 + x_6)\end{aligned} \tag{26}$$

The gradient magnitude and direction are then calculated numerically using the equations

$$\nabla f = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \phi(x,y) = Tan^{-1}\left(\frac{\left(\frac{\partial f}{\partial y}\right)^2}{\left(\frac{\partial f}{\partial x}\right)^2}\right) \tag{27}$$

instead of approximations. The EH is generated in two steps. The first processing sequence calculates the gradient magnitude and direction for each pixel on each color plane, which is stored in memory. The second processing pass examines each stored magnitude and direction. If the gradient *magnitude* is above a threshold, set at 128, then the histogram bin corresponding to the gradient *direction* is incremented. The EH is processed separately with 8 bins for each of the three color planes and concatenated to form the final feature vector.

Initial tests using the EH feature produced mixed results (see Table 4.3 and Appendix C). When used with the Lst color space, the accuracy is the worst of any previously tested feature extractor. The EH RGB extractor, however, is more accurate than the baseline CH Lst extractor, and was therefore selected for use in the system.

| Feature | Resolution | Color Space | Accuracy |
|---|---|---|---|
| Edge Histogram | 8-bin | RGB | 72.47% |
| Edge Histogram | 8-bin | Lst | 65.83% |

Table 4.3 – Full image individual EH classifier results for 4-way test

## 4.3 Combined Feature Extraction and Classification System

The image classification system uses a two-stage feature extraction and classification approach to increase overall recognition accuracy beyond the accuracy of any one of its three constituent classifiers. The original image is presented to the CH, CC, and EH extractors which produce the appropriate feature vectors. Each set of feature vectors is trained and tested independently using three separate SVMs. The output of each SVM is a relative judgment with positive values indicating an indoor image and negative values indicating an outdoor image. For comparison, these three judgments are combined using three different methods to produce the final indoor/outdoor decision.

The first combination method is a simple majority vote. As the system contains three classifiers, ties are impossible, and the final image decision is set to the majority of the classifiers' dispositions. The second combination method simply sums the three distance values from the individual classifiers. Thus, a strong result on one classifier weighs heavier than slight results on the other two and provides a more fine-grained

integration of primary results. The final combination method is a second stage SVM. The three feature extractor outputs are concatenated to form a new feature vector which is used to train another SVM which produces the final image classification.

In addition to functioning on an image as a whole, each feature extractor and first stage SVM supports arbitrary power-of-two image subdivision and classification. For example, the extractors may be configured to subdivide the image into a 4x4 grid, producing 16 subblocks which are classified independently. The 16 resulting classifications are simply summed to produce the overall judgment for the classifier which is then passed to the second stage SVM (see Figure 4.2). The subdivision process is arbitrary and set on a per-extractor basis, so in one system each classifier may function on different subblock configurations simultaneously.

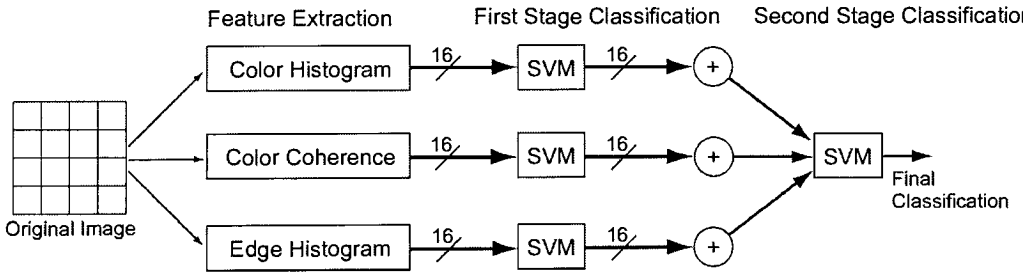



Figure 4.2 – Two stage classification approach

To aid in the parallelization of the feature extraction process, the MPE Upshot profiling and performance analysis tool was utilized to examine the sequential execution of the system (see Figure 4.3). From the plot, the file input phase, CH extractor, and EH histogram generator require the least computational time. The system spends most of its time running the CC extractor and performing the Sobel operator. The CC extractor requires more time than the simpler extractors because each pixel visitation requires

fetching adjacent values corresponding to a mask and calculating coherency with a floating point square root. The Sobel operator takes even more time due to its extensive use of floating point square roots and inverse tangents for the gradient calculations.

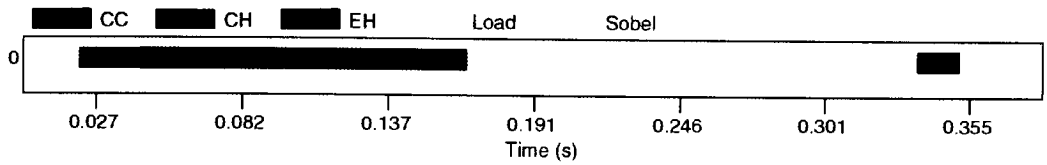



Figure 4.3 – Execution trace before parallelization

The goal of parallelization, with respect to Amdahl's Law, is to optimize the portions of the system which take the most computational time. To this effect, the CH extractor is considered the baseline extractor performance, and additional processors are assigned to the CC and Sobel phases to reduce their time requirements to that of CH. Assigning 4 processors to CC and EH/Sobel roughly balances the computational time, requiring 10 processors for the entire system (see Table 4.4). One processor is used for CH, and one additional processor collects the vectors from the extractors to produce output vector files or operate the SVMs to produce a classification.

| Task | Processors |
|---|---|
| CH | 1 |
| CC | 4 |
| Sobel/EH | 4 |
| Vector Collection and SVM Classification | 1 |

Table 4.4 – Parallel feature extractor processor assignments

Before a parallelization of the feature extraction system may be completed, however, the issue of parallel file input requires attention. In the MPICH workstation cluster environment, all processors have equal access to shared network filesystem (NFS)

volumes. Placing the source image files on a shared NFS volume therefore makes them available to all of the processors, and each extractor may simply open the input files using standard fopen() calls whenever necessary. The filesystem ensures that the data is available to each of the processors when required, and the processors perform the input phase of their operation independently and in parallel.

Unfortunately, the use of NFS volumes for data retrieval introduces a substantial non-deterministic performance variation through filesystem caching. When the images are first accessed, all of the data is present only on the remote server, and NFS must copy the files to all of the other nodes in the cluster on demand. NFS is a particularly inefficient system for cluster operations because it performs its remote lookup operations on a per-machine basis without recognizing that multiple machines in the cluster require the same data at the same time. NFS also caches remote data on the local computer, so additional test runs may exhibit spectacular performance as the image data is retrieved from a local RAM cache instead of a remote disk drive. The use of NFS volumes, therefore, makes a first test run abysmally slow and subsequent executions excessively fast, which is not consistent with the system's anticipated operating characteristics.

To create a more balanced and predictable parallel system, data distribution is performed manually using MPI. The original image file is opened only on the root node in the cluster, which physically contains the disk holding the file. A series of broadcast operations then disseminate the raw image data to each extractor processor. The extractors then send their finalized feature vectors to the collector processor, which writes the output file or performs the classification using the SVM.

## 4.4 Accuracy Results

The feature extractor software was combined with the SVM developed previously to produce the complete classification system. The image classifier is executing using a multi-step process, which requires two runs of the software for training and additional runs to generate output results based on the model. The first run executes only the feature extractors and produces three feature vector files. These three feature vector files must then be processed with a SVM trainer, such as the SVM-SMO msvm product or Joachim's svmLight, to produce the classification models for each extractor. Another run of the system produces a fourth feature vector file, which is similarly trained to create a model for the second stage SVM classifier. When all four models have been generated, the image classification system is ready to examine images and produce results. To describe the accuracy of the system, classifications were performed on the full image as a whole and on a 4x4 subblocked image.

Full Image

Among the three independent feature classifiers, CC RGB produces the most accurate results at 80.83%, followed by EH and CH (see Table 4.5). For the entire system, the best results were produced by the distance classifier, which achieved an accuracy of 83.59%, followed by the second stage SVM and the majority classifier (see Table 4.6 and Appendix D).

| Classifier | 2-way, 8-bin | 4-way, 8-bin |
|---|---|---|
| Color Histogram Lst | 72.01% | 71.50% |
| Color Coherence RGB | 80.83% | 79.65% |
| Edge Histogram RGB | 73.39% | 72.50% |

Table 4.5 – Final full image individual classifier accuracy

| Classifier | 2-way, 8-bin | 4-way, 8-bin |
|---|---|---|
| Majority Classifier | 81.44% | 80.60% |
| Distance Classifier | 83.59% | 82.70% |
| Second Stage SVM | 83.44% | 80.75% |

Table 4.6 – Final full image combined classifier accuracy

Subblocked Image

When the image is subblocked into 16 regions for classification, each block is expected to contain weaker indoor/outdoor characteristics, which reduce the individual classifier accuracy rate. With the 4-way, 8-bin testing suite, this decrease in classification accuracy occurred, as the classifiers dropped from 6% to 8%. However, with the 2-way testing suite, the individual classifiers actually increased in accuracy (see Table 4.7 and Appendix E).

| Classifier | 2-way, 8-bin | 4-way, 8-bin |
|---|---|---|
| Color Histogram Lst | 77.91% | 65.69% |
| Color Coherence RGB | 85.28% | 71.54% |
| Edge Histogram RGB | 78.07% | 63.80% |

Table 4.7 – Final subblocked image individual classifier accuracy

| Classifier | 2-way, 8-bin | 4-way, 8-bin |
|---|---|---|
| Majority Classifier | 86.27% | 85.74% |
| Distance Classifier | 87.19% | 86.61% |
| Second Stage SVM | 87.88% | 86.09% |

Table 4.8 – Final subblocked image combined classifier accuracy

The final results for the entire classification system were produced using the combined extractors operating on a subblocked image. The two-way test provided the best accuracy obtained by the system, 87.88%, achieved with the second stage SVM operating with 8-bin feature vectors (see Table 4.8 and Appendix F). The distance classifier was only slightly less accurate. With the 4-way test, which demonstrates generalization performance, the accuracy diminishes only slightly, but the preferred combination method designation reverses: the best accuracy, 86.61%, was produced with the distance classifier, and the second stage SVM was slightly less accurate.

## 4.5 Performance Results

The parallelization of the feature extraction system produced speedup in both system configurations (see Table 4.9 and Appendix G). Executing sequentially, the feature extractor was capable of processing only 2.32 images per second. The best results were obtained when using the caching filesystem for repeated experimental runs, which produced a processing rate of 12.64 images/s achieving a speedup of 5.44. As expected by the design, the execution time varied substantially due to the caching filesystem's behavior. The implementation with custom input distribution provided a repeatable processing rate of 5.96 images/s for a speedup of 2.57 over the sequential operation.

| Operating Mode | Images/s | Speedup |
|---|---|---|
| Sequential | 2.32 | 1.00 |
| Parallel, np=10, with NFS Cache | 12.64 | 5.44 |
| Parallel, np=10, with Custom I/O | 5.96 | 2.57 |

Table 4.9 – Feature extraction performance

The addition of the first and second stage SVM to the feature extractors to produce the integrated classification system did not result in a noticeable performance drop for the entire-image classifier (see Table 4.10). In fact, the four SVMs increased the processing time for the 326-image 4-way test suite by only 0.09 seconds. Classifying the images using 16 subblocks, however, introduced an additional delay due to the single SVMs operating sequentially. Processing time increased from the 52.41s extraction time to over 368s (6.13m). Surprisingly, the execution time including the second stage SVM is slightly less than the time when the system is run without it. The system without the second stage SVM actually does all of the second stage data processing up to but excluding the SVM classification.

| Execution Stage | Full Image | Subblocked Image |
|---|---|---|
| Extraction only | 52.25s | 52.41s |
| Extraction and stage 1 SVM | 52.33s | 368.68s |
| Extraction and stage 2 SVM | 52.34s | 368.35s |

Table 4.10 – Image classification performance

## 4.6 Discussion

This classification system's accuracy results are consistent with values reported in the literature. Szummer's color histogram and MSAR texture features achieved 90.3% accuracy [15] with some duplicate images and about 85% without [14]. Similarly, Serrano's system achieved 90.2%. Thus, this system's best result of 87.88% is slightly less than these previously reported results. The difference in results is partially attributable to the selection of classifiers and the selection of feature resolution. While

Szummer's histograms utilized 32 bins per channel, and Serrano's 16, this system uses only 8-bin per channel histograms for all feature vectors.

Classifier Accuracy

When subjected to the 2-way tests, the individual color-based classifiers utilized in the combined image processing system were generally as accurate as those used by Serrano. The CH Lst feature achieved accuracies of 72.01% and 77.91% on the individual and subblocked images, respectively, while Serrano's color and texture features produced results of 67.6%. The CC features were even more accurate, achieving an accuracy of 85.28%. The EH texture feature was also as accurate as Serrano's 73.0%, producing results of 73.99% on the full image and 78.07% on the subblocked image.

Application of the four-way test suite indicates that the extractors have mixed generalization performance. The CC extractor remains the most accurate for the full image at 79.65% and the subblocked image at 71.54%. CH and EH similarly drop from 71.50% and 72.50% to 65.69% and 63.80% respectively. The data suggests that EH, originally considered a poor substitute for a texture feature, is about as accurate as CH.

The subblocking operation also affects the performance of the individual feature classifiers. When an image is subblocked and each block is classified independently, the individual classifications are reduced in accuracy because each block has a weaker feature signature [14]. While individual classifications may have less accuracy, the results from the subblocked test demonstrate that the combination of the 16 blocks using a simple arithmetic average produces substantially better results (see Table 4.7, p. 45).

Individual classifier accuracies increase by about 5% when calculated on subblocks instead of the full image.

The combined classifier accuracies for the entire system were surprisingly similar. For both the full image and the subblocked image, the simple three-vote majority classifier was the least accurate, with results at 81.44% and 86.27%. The distance and second stage SVM classifiers produced results exact to the nearest whole percentage: 83% for the whole image and 87% for the subblocked image. As a simple arithmetic summation of first stage results produced high accuracy output, these results suggest that the second stage SVM may not be necessary.

Performance

While the parallelization provided speedup for the feature extraction process, the actual performance gain was less than anticipated. The parallelization itself reduced computational time as expected, but introduced substantial amounts of communication time. The two implementations, one using the shared filesystem and the other MPI for initial image data distribution, have strengths and weaknesses which depend on the desired application.

The filesystem distribution method relies on the shared network filesystem to produce the raw input data files for each of the processors upon demand. Unfortunately, the system's performance on original images was completely unacceptable. Whenever the software was executed with images that had not been processed since a cluster reboot, the filesystem required an excessive amount of time to copy the images to all of the workstations and performance suffered dramatically. After the first execution, the images

would be cached locally, so subsequent runs proceeded extremely quickly as the files were loaded from the NFS RAM cache. While the RAM cache performance was very fast, its behavior relies on preloading data to remote workstations, and this implementation was not considered for performance measurements.

The other implementation, which uses explicit MPI broadcast calls to distribute the raw image data throughout the cluster, produced reproducible and consistent timing results. Because the images must be transmitted over the network every time the program is run, this implementation is slower than the NFS cache, but this is characteristic of the execution expected in a system classifying completely new images in a dynamic run-time environment. The expected side effect of parallelization, a mandatory introduction of communication time, reduces performance as each processor must wait for data loaded from the root node, which shows up as whitespace in the Upshot trace (see Figure 4.4).

Due to its reliance on the cache, the filesystem-based distribution method is actually beneficial for repeated test runs that process the same images, but is unrealistic for an actual classification system. Substantial portions of feature extractor design and SVM classifier testing involve generating several slightly different feature vector sets based on the same images. In such an experimental setup, the implementation using the cache will provide the better performance. Further design considerations must identify the system's actual operating characteristics for optimization as an experimental or a production system.

The cluster used to execute the feature extractor provided no inherent parallel input or output functionality. Images were merely read from a single consumer-grade IDE disk in a desktop computer designated the NFS server. Slightly better performance

would probably be provided by a server or workstation class computer with SCSI drives. Similarly, the use of a storage area network (SAN) device natively implementing NFS would improve performance. In this system, one computer was required to transmit the data to the other computers, by the filesystem or with MPI. A SAN solution designed for fast parallel data distribution would place all processors on an equitable input/output platform and completely eliminate the data distribution problems. Finally, in a production environment such as a photographic processing unit, images may not even be retrieved from disk! Therefore, the data distribution method is of fundamental importance, and for a more complete system the most important design consideration must identify the source of the images and how these images are transmitted to the nodes in the cluster.

While the file distribution proved problematic, the parallelization of the processing portion of the system produced a balanced execution profile between the nodes in the cluster (see Figure 4.4). The CH and Sobel operations, which occupied the most time in the sequential system, were successfully distributed across four processors. From the Upshot plot, each node in the feature extractor performs roughly the same amount of work.

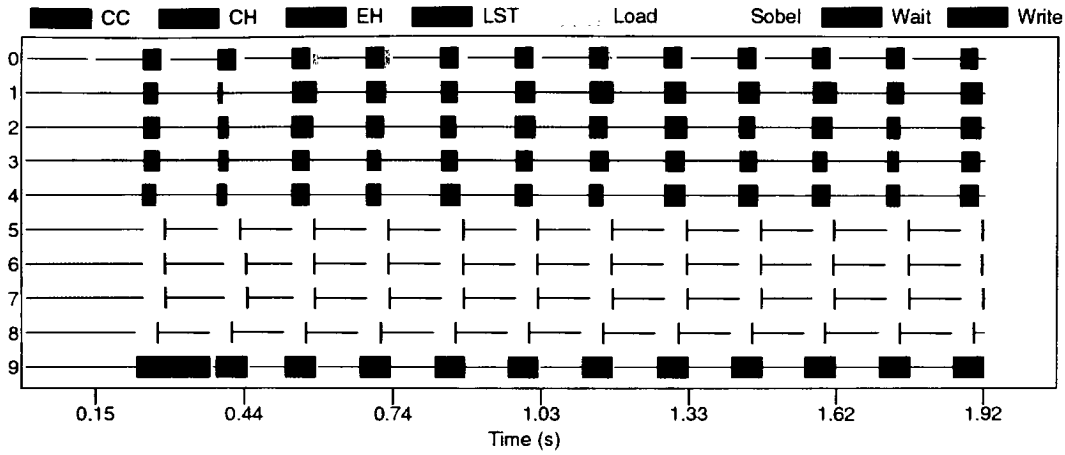



Figure 4.4 – Execution trace after parallelization

Feature extraction constitutes only a portion of the classification system's work. The inclusion of the SVM classification stage also affected the performance of the system, and demonstrated one possible design flaw in this parallel extraction and classification system. In the case of the full image system, the classification component requires negligible computational time. For one test set, both extraction and classification could be performed in about 52 seconds. With subblocked images, however, the SVM classification stage introduces a significant delay (see Table 4.10, p. 47). Running the same test set requires only 52 seconds for extraction, but over 6 minutes when combined with the SVM.

This increase in processing time is due to two factors. First, the subblocked method requires classifying more feature vectors for each image. For the full image test, only 3 vectors must be classified: CC, CH, and EH. With the subblocked method, each extractor produces 16 vectors, for a total of 48 vectors requiring classification for one image. Second, the subblocked SVM models require substantially more time to execute.

In a nonlinear SVM, an output evaluation requires computing a dot product between the feature vector and every SV in the model. But because the subblocked image models were trained with subblocked samples, they contain many more SVs than do the models generated using full image vectors (see Table 4.11).

| Model for 4-way test 1 | Full Image | Subblocked Image |
|---|---|---|
| Color Histogram | 228 | 3839 |
| Color Coherence | 201 | 3352 |
| Edge Histogram | 263 | 4189 |

Table 4.11 – Number of SVs in feature classifier models

The increase in model SVs dramatically decreases performance. Not only does the subblocked approach require classifying more vectors, but each classification takes more time. For example, the CC classifier for the full image can produce 861.1 classifications/s. The larger model file of the CC extractor for the subblocked image can only produce 56.9 classifications/s. The other classifiers have similar performances. Requiring 48 classifications translates into about 0.84s of computer time for each image, which results in the increase in execution time for the subblocked case.

This additional processing requirement is not unreasonable in and of itself, but presents a problem to this particular implementation due to one design decision: the SVM classification stage is implemented sequentially on one processor. This point of contention is necessary to provide the software's flexibility in the subdivision of images for feature extraction. That is, one processor may be used to create a single vector or 16 vectors, or 4 processors may be used to create a single vector or 16 vectors. In the case of the 4 processor to 1 vector relation, it is necessary for the four vectors to be summed on one machine before classification.

Retrospectively, moving all of the feature vectors to one node for classification was not an incorrect decision. Indeed, all of the vectors must be moved to one node to be output in a text file for use with training the SVM model, and attaching the SVM classifier to the same node was a logical extension of the data flow. However, in the case of the subblocked configuration, classifying the images on this node has shown to be inefficient.

The extraction may be implemented in parallel in several different ways. First, three additional processors could be allocated to perform the classifications in a pipelined fashion. The data would be collected from the extractors, passed to the classifiers, and then sent to the final node to produce the result. Second, the classifiers could be integrated with the extractors. After extraction is complete, the vectors could be collected on the first processor running the extractors and classified locally, with the results transmitted to the collection node. Furthermore, either parallelization of the extraction would require consideration for the distribution of the SVM models from disk during program initialization. Either change, however, would represent a substantial deviation from the system's current design and workflow and require significant software redevelopment.

Overall, the parallel feature extractor provided performance gains while achieving accuracy results just slightly less than recent reports from the literature. In addition, the feature extraction system supports runtime flexibility in the selection of color model, histogram bin size, and image partitioning through subblocking. The software provides a completely integrated, extraction-to-classification system capable of producing indoor/outdoor classifications from raw image data.

Chapter 5

# Electronic Mail Classification

Unsolicited commercial electronic mail (UCE), frequently referred to as spam, has plagued Internet users for almost an entire decade. The first recognized incident of large-scale mass mailing occurred in 1994, when two California attorneys sent their green card solicitation to most of USENET [3]. Since that time, bulk mailing software has proliferated, making the transmission of thousands messages with forged headers and falsified contact information possible without much technical knowledge.

Unwanted e-mail is generally categorized under the blanket term spam, although a casual analysis suggests that multiple types exist. It is particularly important to distinguish between solicited commercial e-mail and UCE. Solicited commercial e-mail is typically generated by a large corporation in response to a request for additional periodic information. For example, many organizations and companies publish occasional newsletters, such as a summary of journal articles, full-text publications, or even monthly compact disc recommendations. Because they are authored by corporations which tend to be cognizant of their customers' privacy and complaints, solicited commercial e-mail can typically be abated through an unsubscribe process. Occasionally, corporate e-mail may be solicited for delivery to a third party recipient. These messages, such as online greeting card notifications and "tell a friend about this page" referral services, generate a nebulous form of UCE. As these messages are typically generated by a database server, they contain no personal content yet are

solicited. True spam, however, generally lacks any corporate legitimacy. According to Cranor's 1994 analysis [3], spam advertises a wide range of products and services including computer hardware and software, office products and services, adult entertainment products, make-money-fast (MMF) opportunities and pyramid schemes, as well as so-called business opportunities via mass e-mail marketing with software and mailing lists. Because of the complaints generated by spam, senders typically falsify header information to direct complaints to nonexistent or uninvolved third-party accounts. Addresses are collected through scanning USENET and published web documents, and options to unsubscribe are either missing or advertised but ignored.

In the past, spam filtering at the server level required the manual construction of pattern matching rule sets [3]. This process was not particularly efficient, catching only a small portion of incoming UCE and requiring extensive amounts of time. Probabilistic classification techniques simplify the filtering process by providing automated classification model generation based on representative samples. The aim of this chapter is to use a SVM to construct an automated classification system to detect unsolicited commercial e-mail.

## 5.1 Data Sets

For the purposes of training and testing the classification SVM, several sets of sample electronic mail were collected (see Appendix I). Over a two month period, individuals at RIT's Information Technology Services help desk archived 1342 messages which were evaluated by the recipient as junk mail. These messages were combined with 1342 nonspam personal messages collected by the author over a period of two years. The

combined e-mail data set, which contained 2684 messages, was randomly divided into a training set and a testing set containing 1340 and 1344 messages, respectively, each with an equal number of spam and nonspam messages. This set is referred to as Set A.

To assist in evaluating the generalization performance of the classifier, two additional data sets were collected and used for testing only. The second set, referred to as Set B, contains 77 spam messages collected by the author. The third set, Set C, contains 378 nonspam messages collected by another individual. Sets B and C are intended to examine if a filter built using other people's training data can be applied impersonally, that is, without building a specific model for a particular recipient.

## 5.2 Method

In text categorization, a feature is a word. Each mail message was parsed to completely remove any headers, attachments, HTML markup, punctuation, and extended characters. This procedure essentially reduces a mail message to a series of delimited lowercase string tokens. The messages in training set A were used to generate a raw list containing the 19,108 tokens found in the messages. In addition to legitimate words, inline hyperlink URLs, image URLs, mailto directives, and occasional mistyped HTML tags become tokens in the list as well.

The raw list was processed by removing any token which occurred less than three times in the sample set. In addition, any tokens beginning with http or mailto, and tokens consisting entirely of numeric digits, were removed. This resulted in a dictionary of 5,841 words. Because the dictionary was generated by an analysis of the personal e-mail of an individual, a significant number of proper nouns were present. This dictionary is referred

to as the personal dictionary. As an additional test, the 16 most frequent proper nouns were removed, resulting in a more impersonal dictionary. The removal of proper nouns prevents classification based on names used as closing lines or signatures, forcing the classifier to examine only generic words. Finally, the system wordlist retrieved from \usr\dict was utilized as an alternate dictionary.

Feature vectors using the three dictionaries were generated from the training set. Unfortunately, a small percentage of e-mail messages did not contain any words in the dictionary. These messages cannot be parsed to generate feature vectors, and are omitted from the subsequent testing results. This rate of omitted messages is typically very small, ranging from 1% to 7% (see Appendix I), depending on the contents of the data set. The feature vectors were used to produce a classification model using the SVM-SMO product developed previously. With the SVM parameter C set to 10 and a linear kernel, the training time for each model was quite short, requiring less than 2 minutes each. All three models produced valid solutions with perfect recall accuracies, which were then used to test the remaining data sets.

## 5.3 Results

The electronic mail classification models were quite accurate with results exceeding 96% (see Table 5.1 and Appendix J). The personal dictionary produced the system's best results with an accuracy of 96.69% on Set A and 97.33% on Set B. The impersonal dictionary produced the next best results, which was only 1.33% lower on the Set B test. The system dictionary produced the least accurate classification results for all test sets.

| Test Set | Personal Dictionary | Impersonal Dictionary | System Dictionary |
|---|---|---|---|
| Set A | 96.69% | 96.69% | 95.26% |
| Set B All Spam | 97.33% | 96.00% | 89.33% |
| Set C Subset | 96.26% | 96.55% | 93.10% |
| Set C Complete | 93.58% | 93.85% | 93.10% |

Table 5.1 – Mail Classification Accuracy

The results for Set C were particularly lower than the other test sets at about 93% on all dictionaries. An analysis of the incorrect classifications in Set C indicated that the data set contained 1 spam message forwarded to the recipient by a friend with a short sarcastic comment and 9 automatically generated electronic greeting card reminders. These messages, generated by a server, contained no personal content but large disclaimers and a single unique hyperlink to retrieve the card. Removing these 10 messages from Set C, producing the Set C Subset, increased the accuracy to above 96% consistent with the other test sets.

## 5.4 Discussion

Analysis of the linear SVM models clearly indicates which dictionary words are associated with spam and nonspam messages as well as which words are not significant in determining the classification (see Table 5.2 and Appendix K). The personal and impersonal dictionaries, generated through a dictionary-building process, resulted in words which were roughly balanced between nonspam, neutral, and spam categories. The system dictionary, however, had the fewest number of words in its classification categories, which may explain its lesser performance.

| Dictionary | Raw Words | Nonspam | | Neutral | | Spam | |
|---|---|---|---|---|---|---|---|
| Personal | 5841 | 1888 | 32.32% | 2172 | 37.18% | 1781 | 30.49% |
| Impersonal | 5825 | 1838 | 31.86% | 2131 | 36.58% | 1856 | 31.55% |
| System | 25473 | 1192 | 4.67% | 23331 | 91.59% | 950 | 3.72% |

Table 5.2 – Trained Dictionary Statistics

The personal dictionary, as expected, produced a model with large nonspam weights for proper nouns associated with the author. These nouns included both first and last name, academic institution, and the names of roommates. When these names were removed from the dictionary, and were therefore not available for classification, accuracy did not change substantially. The results for Set A remained the same, Set B had one additional incorrect classification, and Set C had one additional correct classification. These results suggest that a personalized dictionary including proper nouns may not be particularly important and that a well-developed generic dictionary may be applied at the server level to an incoming mail stream.

Tests using the generic system dictionary had substantially poorer results than the dictionaries generated by analyzing the training set. While the system dictionary contained 4.3 times as many words as the generated dictionary, fewer words were assigned non-neutral classification weights. A cursory analysis suggests that this may be because the official dictionary contains only real words, and that several expletives and slang terms useful in distinguishing spam from nonspam mail are not present in the dictionary. Feature vector generation with the system dictionary is also slower, as the parser and counter must scan through the larger list to determine which words are present.

While the selection of words is important, the presence of words is essential. In order for the SVM to be able to generate a classification, the message must first be parsed

into tokens and correlated with the dictionary to produce a feature vector. This seems intuitive, but electronic mail messages do not need to contain text. In the sample sets used for the training and testing of this classifier, several messages were discarded because they contained no words which were present in the dictionary. These messages may fall into three categories. First, the message may be blank, with the entire content of the transmission included in the subject header. This system considered only the body of the message, but in future work, prepending the subject to the body for feature extraction may prove beneficial. Second, the message may actually not contain any words. These messages are typically generated by the sender intentionally pasting a URL into an e-mail client with no additional conversation. Third, the message may not contain any words due to image-only HTML markup. While the previous two unclassifiable message types may be spam or nonspam, this final category is a feature generally only present in junk mail. Individuals typically type messages directly into an e-mail client, but some spam authors present their advertisement only as JPG or GIF graphics referenced using HTML image tags. The feature extraction parsing sequence eliminates all HTML tags, which leaves the document blank. This type of message is particularly problematic, as the transmission contains no ASCII text except for the headers, which may be used to produce an automated classification.

Although the classification system obtained a high accuracy, it remains far from perfect. Erroneously classifying a junk message as legitimate, referred to as a miss, is generally acceptable behavior because the message can be examined and easily deleted. Misclassifying a nonspam message, known as a false alarm, is unacceptable if the message is permanently deleted without human interaction. The classification system

with Set A had a miss rate of only 2.10% and a false alarm rate of 4.52%. This means that 2.10% of all messages labeled as nonspam were actually junk mail, and that 4.52% of all messages labeled as spam were indeed legitimate. For this reason, it is important to use the mail classification system as a labeling and prioritizing assistant only. Instead of automatically deleting messages categorized as spam, messages should be processed in a manner which allows later human interaction. For example, spam messages may be moved to another folder, moved to the bottom of the inbox, or flagged with a particular color.

With a reported miss rate of 4.52%, it may appear that the system is blatantly and overtly, almost antagonistically, misclassifying personal messages as spam. This is not the case. The misclassifications are generally due to solicited commercial messages which contain many of the features of spam, including automatic generation, lack of personalized content, and large repeated unsubscribe notices. Ultimately, one person's spam may be another person's subscription, so a certain genre of messages will require manual intervention either interactively or through the construction of traditional processing rules.

## 5.5 Microsoft Outlook Integrated Spam Scanner

To provide the benefits of SVM-based e-mail classification to ordinary end users, an add-in DLL for Microsoft Outlook XP was developed. This software examines electronic mail messages on delivery, running each message delivered to the default Inbox through feature extraction and SVM classification stages. Written entirely in

Microsoft Visual Basic, the spam scanner add-in implements the extractor and SVM internally, and interfaces with Outlook XP using its published object model.

When new mail is received, the spam scanner automatically loads and begins the classification process. A progress dialog is displayed modally to keep the user informed of the operation's progress (see Appendix L). As each message is classified, the number of words is displayed with the final classification. Spam messages are designated with a red X icon, while nonspam messages appear with a green check. When processing completes, the dialog box closes automatically. The classification process for two dozen messages on a Pentium III-500 requires only a few seconds.

The results of the spam classification are stored with the Microsoft Outlook mailbox item using a custom property named SPAM. This custom property becomes a field associated with the information store, so it is available to the Outlook environment using the traditional user interface. Thus, the SPAM field may be displayed as a column in the view of e-mail messages and used as criteria for conditional formatting. For example, messages that are labeled spam may be displayed in a different color than the legitimate messages (see Appendix M).

The electronic mail classification application provides a highly efficient and accurate method of prescreening unsolicited e-mail. Without generating any custom rules, such as exceptions for preferred subscriptions, the system's accuracy exceeded 96%. In addition to the experimental setup utilized for generating and testing the models, the Microsoft Outlook add-in allows ordinary users to utilize SVM technology to find and discard mailbox junk.

# Chapter 6

# Conclusions

## 6.1 Summary of Work

This thesis examined the development and utilization of SVMs through the production of several software applications (see Appendix N). The first stage focused on a sequential SVM-SMO trainer, which was compared to results produced by other SMO and non-SMO packages to verify the system's accuracy. After completing the sequential implementation, two parallelization techniques were examined. One technique, utilizing interleaved parallelization, was discarded due to difficulties in reaching valid termination conditions. The other technique, using a larger-grained blocked parallelization, produced substantial speedup but violated the KKT conditions therefore reducing the trainer's accuracy. The sequential trainer was used to generate the models used throughout the remainder of the thesis.

The second component of work developed a parallel feature extraction system. Given raw image files, the extractor uses 10 processors to extract color histogram, color coherence, and edge histogram features for each image. The system then implements a SVM evaluator which may be used to label the images as indoor or outdoor. The parallel feature extraction system demonstrated speedup of 2.57 and 5.44, depending on whether the data was initially distributed manually or through the cached network file system.

The third segment of the thesis project developed an electronic mail classification system capable of distinguishing between spam and nonspam messages with over 96%

accuracy. Implemented with substantial code reuse through integrating the previously developed feature extractor and SVM system, the e-mail classification system was developed in less than one week. Finally, the extraction and SVM classification algorithms were ported to Microsoft Visual Basic and encapsulated in a DLL for integration with Microsoft Outlook XP, bringing SVM-based mail classification to the desktop computer environment.

## 6.2 Challenges Encountered

Each component of the thesis project presented various challenges and difficulties, most of which were overcome to produce the final product and reported results. The most challenging portion of the work involved the original implementation of the SMO training algorithm. SVM training with SMO is a nontrivial algorithm which requires considerable attention to detail in order to implement properly. The most difficult portion of the project, however, was the parallel implementation of the SMO trainer. While the final results produced speedup with tolerable loss in accuracy, the solution technically violates the KKT conditions and is dependent on the distribution of data. As such, the implementation is not deterministic and unreliable for essential applications. This non-optimal solution was the result of substantial experimentation and additional research, but is not the neatly implemented parallel algorithm translation originally anticipated. The SMO algorithm does not appear to be an appropriate choice for parallelization because of its reliance on a global bias parameter. In retrospect, proposing to develop a parallel version of an algorithm named *sequential* minimal optimization seems counterintuitive.

Compared to the SMO algorithm stages, the remainder of the thesis project presented no major obstacles or difficulties. In the image classification component, designing an efficient methodology for original data distribution through the cluster required experimentation although the result was extraordinarily simple. The electronic mail classification component was practically an exercise in repetition: the feature extraction code from the image classifier and the SVM evaluation engine from the trainer were combined with changes to produce the final product. Thus, the final stages of the thesis project proceeded quickly and with minimal redesign, as each subsequent project successfully built on previous work.

## 6.3 Future Work

The three product components developed in this thesis have presented several opportunities for future work. The first and most obvious potential topic for further research remains the development of a parallel trainer for SVMs. Not a simple implementation issue as originally presumed, the creation of a parallel SVM trainer would require extensive theoretical work in QP solving and numeric methods. A viable parallel solution would probably be based on a matrix-based numeric solution, such as Osuna's algorithm, instead of SMO.

The parallel image classification project currently functions as a run-time system on a Linux cluster, but several issues need to be addressed. Prior to additional development, the system's intended image distribution method must be clearly specified. Identifying the source of the images for classification, as well as selecting whether to use

full images or subblocked images for classification, is essential in selecting the appropriate parallelization strategy for high performance.

Last, the Microsoft Outlook spam scanner product may be suitable for commercial development after the addition several essential features. First, the software's scanning flexibility should be enhanced, allowing folders in addition to the Inbox to be scanned on demand. Second, an automated model download feature, which would occasionally retrieve automatic updates of the SVM model, must be implemented to maintain currency and sustain filtration accuracy over time. Finally, the software should be modified to examine the user's Contacts folder and grant immediate acceptance to messages from recognized acquaintances without even scanning the content, greatly reducing the chance that a desired message is classified as spam.

Client side scanning is only one opportunity to catch unsolicited commercial e-mail. With additional development, the SVM e-mail scanner may be useful in both incoming servers and outgoing mail (SMTP) servers. For example, a Microsoft Exchange Server add-in or a modified version of the UNIX command line mail scanner could be integrated with mail servers to discard or categorize incoming spam at the server level. Internet service providers could also implement rudimentary spam categorization on their outbound SMTP servers. At the strictest implementation, messages generating high spam scores could be returned to the sender without being transmitted over the Internet. A less intrusive implementation could monitor outbound SMTP traffic for excessive volumes of potential spam and alert an administrator to investigate manually when a user may be violating the acceptable use policies. Regardless of its location, e-mail classification demonstrates a successful application of SVMs.

Despite their power and versatility, Platt noticed that "SVMs have not yet enjoyed widespread adoption in the engineering community," possibly due to the complexity, subtlety, and implementation difficulties [12, p. 1]. While SVMs certainly require attention to detail, SVM training needs to be performed only once, and the trainer need not be written from scratch. The practical benefit of SVM technology is the actual document classification task, which can be easily implemented in languages such as C and Visual Basic. Overall, SVMs provide an efficient and accurate algorithm for a variety of automated classification tasks, as evidenced by the results of this thesis.

## BIBLIOGRAPHY


1. deAlmeida, M. B. "SMO – Sequential Minimal Optimization." Computer Source Code, 2001. http://www.cpdee.ufmg.br/~barros.

2. Burges, C. "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery*, 2, 1-43, 1998.

3. Cranor, L. F., LaMacchia, B. A. "Spam!" *Communications of the ACM*, 41, 74-83, 1998.

4. Drucker, H., Wu, D., Vapnik, V. N. "Support Vector Machines for Spam Categorization." *IEEE Transactions on Neural Networks,* 10, 1048-1054, 1999.

5. Etin, S., Elias, U. "Parallelizing SMO for Solving SVMs." Unpublished manuscript, Technion – Israel Institute of Technology Vision Research and Image Science Laboratory, 2001. http://www-visl.technion.ac.il/visl/projects/2000/24w99/documents/book/.

6. Ge, X. "Sequential Minimal Optimization for SVM." Unpublished Manuscript, University of California, Irvine, 2001. http://www.ics.uci.edu /~xge/svm/.

7. Hearst, M. "Support Vector Machines." *IEEE Intelligent Systems*, 18-28, July 1998.

8. Joachims, T. "Text Categorization with Support Vector Machines: Learning With Many Relevant Features." *Machine Learning: ECML-98, Tenth European Conference on Machine Learning*, 137-142, 1997. http://citeseer.nj.nec.com/joachims98text.html

9. Kwok, T. "Automated Text Categorization Using Support Vector Machines." *Proceedings of the International Conference on Neural Information Processing, Kitakyushu, Japan*, 347-351, 1998. http://www.comp.hkbu.edu.hk/~jamesk/papers/iconip98.ps.gz.

10. Muller, S. "High Performance Hardware Acceleration for Image Database Organization, Browsing, and Retrieval." Thesis Manuscript, Department of Computer Engineering, Rochester Institute of Technology, 2002.

11. Osuna, E., Freund, R., and Girosi, F. "Support Vector Machines: Training and Applications." MIT Artificial Intelligence Laboratory, 1997. ftp://publications.ai.mit.edu/ai-publications/1500-1999/AIM-1602.ps.Z.

12. Platt, J. "Sequential Minimal Optimization: a Fast Algorithm for Training Support Vector Machines." Microsoft Research Technical Report MSR-TR-98-14, 1998. http://www.research.microsoft.com/~jplatt/smo.html.

13. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. "A Bayesian Approach to Filtering Junk E-mail." *AAAI-98 Workshop on Learning for Text Categorization*, 1998. http://research.microsoft.com/~sdumais/spam98.ps.

14. Serrano, N., Savakis, A., Luo, J. "A Computationally Efficient Approach to Indoor/Outdoor Scene Classification." *International Conference on Pattern Recognition, Montreal, Canada*, August 2002.

15. Szummer, M., Picard, R. "Indoor-Outdoor Image Classification." In *Workshop in Content-based Access to Image and Video Databases, Bombay, India,* 1998. ftp://whitechapel.media.mit.edu/pub/tech-reports/TR-445.ps.Z.

16. Wilkinson, B., Allen, M. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers.* Prentice Hall, 1999.

17. Yang, Y. "An Evaluation of Statistical Approaches to Text Categorization." *Journal of Information Retrieval*, 1, 67 – 88, 1999.

## APPENDIX A – SVM SOFTWARE OPERATION

### General Command Line Format

msvm –train|-test –i inputfile –o outputfile –m modelfile
[–C *parameter-C* ] [–e *epsilon*] [–tol *tolerance*]
[–kernel rbf | –kernel linear] [–sigma *sigma*]

**Action Selection**

-train — Run the SVM in training mode. The support vectors in the input file are used to train a new SVM.

-test — Run the SVM in testing mode. Each sample in the input file is classified and the results recorded in the output file.

**File Options**

-i *filename* — Input filename. This file contains the sample vectors that will be used for the training or testing operation, and may contain a dense matrix, sparse vectors, or sparse binary vectors as described by the file specification below.

-o *filename* — Output filename. Each vector in the input file is classified using the SVM and the results are recorded in this file.

-m *filename* — Model filename. In training mode, this file contains the new SVM model. For testing mode, the model in this file is loaded and used to generate new classifications.

**SVM Parameters**

-C *parameter-C* — SVM parameter C, the maximum value for a LaGrange multiplier. Default value 5, typical values include 1 and .05.

-e *epsilon* — Epsilon tolerance for bound precision calculations, default .001.

-tol *tolerance* — Tolerance for satisfying KKT conditions, default value .001.

-kernel linear — Specify linear kernel (default):

$$(x_1 \cdot x_2)$$

-kernel rbf
-sigma *value* — Specify radial basis function kernel:

$$\frac{-\left(|x_1 - x_2|^2\right)}{\sigma}$$

Note that some SVM systems use σ evaluated through the denominator ($2\sigma^2$).

**Testing Set Summary**

Complete training set results for all tested data sets are included on the enclosed CD-ROM. The exact command lines used to generate the results are also included as a set of shell command files. The parameters used with each test set are summarized below:

| Test | Kernel | C |
|---|---|---|
| Adult | Linear | .05 |
| Adult | RBF, sigma = 20 | 1 |
| Tic Tac Toe | Linear | 1 |
| Tic Tac Toe | RBF, sigma = 2 | 1 |
| Pima | RBF, sigma = 2 | 1 |

## APPENDIX B – PARALLEL SVM RESULTS

These tables describe the accuracy and performance for the SVM operating in sequential and parallel modes of operation. The reported accuracy is the self-test recall percentage, produced by testing the SVM with the training set, which is used to quickly gauge the change in accuracy between the different implementations.

Training Times for Adult Data Set, Linear Kernel

| Data Set | | Sequential | | | Parallel x 2 Processors | | | Parallel x 4 Processors | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Set | Samples | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s |
| Adult1 | 1605 | 16.1 | 0.8461 | 84.1 | 6.7 | 0.8312 | 199.3 | 4.8 | 0.8187 | 272.9 |
| Adult2 | 2265 | 29.6 | 0.8340 | 63.7 | 11.8 | 0.8256 | 158.5 | 8.2 | 0.8079 | 224.4 |
| Adult3 | 3185 | 50.7 | 0.8440 | 53.0 | 18.9 | 0.8295 | 139.7 | 13.0 | 0.8053 | 196.8 |
| Adult4 | 4781 | 99.4 | 0.8446 | 40.6 | 36.8 | 0.8297 | 107.8 | 19.4 | 0.8095 | 199.4 |
| Adult5 | 6414 | 168.5 | 0.8481 | 32.3 | 64.0 | 0.8340 | 83.5 | 29.1 | 0.8157 | 179.5 |
| Adult6 | 11220 | 412.0 | 0.8462 | 23.0 | 143.8 | 0.8284 | 64.7 | 66.4 | 0.8037 | 135.8 |
| Adult7 | 16100 | 824.5 | 0.8455 | 16.5 | 250.5 | 0.8241 | 53.0 | 104.4 | 0.7916 | 122.0 |
| Adult8 | 22696 | 1648.2 | 0.8455 | 11.6 | 525.7 | 0.8257 | 35.6 | 187.7 | 0.8025 | 97.0 |
| Adult | 32561 | 3068.8 | 0.8477 | 9.0 | 948.8 | 0.8263 | 28.4 | 307.9 | 0.7972 | 84.3 |

Training Times for Adult Data Set, RBF Kernel

| Data Set | | Sequential | | | Parallel x 2 Processors | | | Parallel x 4 Processors | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Set | Samples | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s |
| Adult1 | 1605 | 489 | 0.8579 | 2.81 | 134 | 0.8480 | 10.15 | 36 | 0.8343 | 36.48 |
| Adult2 | 2265 | 1114 | 0.8494 | 1.73 | 321 | 0.8313 | 5.85 | 80 | 0.8128 | 22.84 |
| Adult3 | 3185 | 2631 | 0.8565 | 1.04 | 566 | 0.8408 | 4.73 | 175 | 0.8380 | 15.17 |
| Adult4 | 4781 | 6054 | 0.8536 | 0.67 | 1170 | 0.8446 | 3.45 | 358 | 0.8369 | 11.18 |
| Adult5 | 6414 | 10929 | 0.8570 | 0.50 | 2591 | 0.8480 | 2.10 | 572 | 0.7749 | 8.69 |
| Adult6 | 11220 | 40203 | 0.8548 | 0.24 | 9057 | 0.8511 | 1.05 | 2378 | 0.8411 | 3.97 |
| Adult7 | 16100 | 111646 | 0.8562 | 0.12 | 20724 | 0.8468 | 0.66 | 5789 | 0.8458 | 2.35 |

Training Times for Tic Tac Toe Data Set

| Data Set | | Sequential | | | Parallel x 2 Processors | | | Parallel x 4 Processors | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Kernel | Samples | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s |
| Linear | 958 | 158.9 | 0.98 | 5.9 | 72.9 | 0.98 | 12.9 | 41.9 | 0.98 | 22.5 |
| RBF | 958 | 1389.8 | 1.00 | 0.7 | 542.0 | 0.98 | 1.7 | 250.5 | 0.92 | 3.5 |

Training Times for Pima Set

| Data Set | | Sequential | | | Parallel x 2 Processors | | | Parallel x 4 Processors | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Kernel | Samples | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s | Time (s) | Recall | Rows/s |
| RBF | 768 | 230 | 1 | 3.33 | 168.21 | 1 | 4.56 | 109.61 | 1 | 7.01 |

## APPENDIX C –FULL IMAGE INDIVIDUAL CLASSIFIER RESULTS

These tables describe the SVM classification performance for the three image classifiers used for indoor/outdoor categorization. The results from color histogram, color coherence, and edge histogram classifiers operating on the full image independently are presented for several color space and bin variants. For all final accuracy results, the self-test recall values are omitted.

C = Correct, I = Incorrect, A = Accuracy

Full Image 4-way, 8-bin RGB Color Coherence

| Testing Set | cc08rgb-4-1 | | cc08rgb-4-2 | | cc08rgb-4-3 | | cc08rgb-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| cc08rgb-4-1 | 265 | 61 | 261 | 65 | 255 | 71 | 252 | 74 | 768 | 210 | 0.7853 |
| cc08rgb-4-2 | 255 | 71 | 275 | 51 | 263 | 63 | 256 | 70 | 774 | 204 | 0.7914 |
| cc08rgb-4-3 | 259 | 67 | 260 | 66 | 270 | 56 | 258 | 68 | 777 | 201 | 0.7945 |
| cc08rgb-4-4 | 259 | 67 | 266 | 60 | 272 | 54 | 269 | 57 | 797 | 181 | 0.8149 |
| **Total** | | | | | | | | | **3116** | **796** | **0.7965** |

Full Image 4-way, 8-bin Lst Color Coherence

| Testing Set | cc08lst-4-1 | | cc08lst-4-2 | | cc08lst-4-3 | | cc08lst-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| cc08lst-4-1 | 267 | 59 | 255 | 71 | 266 | 60 | 280 | 66 | 781 | 197 | 0.7986 |
| cc08lst-4-2 | 260 | 66 | 262 | 64 | 262 | 64 | 252 | 74 | 774 | 204 | 0.7914 |
| cc08lst-4-3 | 253 | 73 | 247 | 79 | 271 | 55 | 258 | 68 | 758 | 220 | 0.7751 |
| cc08lst-4-4 | 252 | 74 | 255 | 71 | 261 | 65 | 273 | 53 | 768 | 210 | 0.7853 |
| **Total** | | | | | | | | | **3081** | **831** | **0.7876** |

## APPENDIX C – FULL IMAGE INDIVIDUAL CLASSIFIER RESULTS

C = Correct, I = Incorrect, A = Accuracy

Full Image, 4-way, 16-bin Lst Color Histogram

| Testing Set / Training Set | ch16lst-4-1 | | ch16lst-4-2 | | ch16lst-4-3 | | ch16lst-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | I | C | I | C | I | C | I | C | I | A |
| ch16lst-4-1 | 247 | 79 | 227 | 99 | 233 | 93 | 251 | 75 | 711 | 267 | 0.7270 |
| ch16lst-4-2 | 224 | 102 | 255 | 71 | 238 | 88 | 241 | 85 | 703 | 275 | 0.7188 |
| ch16lst-4-3 | 228 | 98 | 237 | 89 | 257 | 69 | 245 | 81 | 710 | 268 | 0.7260 |
| ch16lst-4-4 | 232 | 94 | 223 | 103 | 239 | 87 | 259 | 67 | 694 | 284 | 0.7096 |
| **Total** | | | | | | | | | **2818** | **1094** | **.7203** |

Full Image, 4-way, 8-bin Lst Color Histogram

| Testing Set / Training Set | ch08lst-4-1 | | ch08lst-4-2 | | ch08lst-4-3 | | ch08lst-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | I | C | I | C | I | C | I | C | I | A |
| ch08lst-4-1 | 245 | 81 | 225 | 101 | 237 | 89 | 238 | 88 | 700 | 278 | 0.7157 |
| ch08lst-4-2 | 221 | 105 | 251 | 75 | 239 | 87 | 236 | 90 | 696 | 282 | 0.7117 |
| ch08lst-4-3 | 232 | 94 | 229 | 97 | 252 | 74 | 236 | 90 | 697 | 281 | 0.7127 |
| ch08lst-4-4 | 230 | 96 | 232 | 94 | 242 | 84 | 257 | 69 | 704 | 274 | 0.7198 |
| **Total** | | | | | | | | | **2797** | **1115** | **0.7150** |

Full Image, 4-way, 16-bin RGB Color Histogram

| Testing Set / Training Set | ch16rgb-4-1 | | ch16rgb-4-2 | | ch16rgb-4-3 | | ch16rgb-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | I | C | I | C | I | C | I | C | I | A |
| ch16rgb-4-1 | 255 | 71 | 237 | 89 | 245 | 81 | 230 | 96 | 712 | 266 | 0.7280 |
| ch16rgb-4-2 | 217 | 109 | 268 | 58 | 246 | 80 | 233 | 93 | 696 | 282 | 0.7117 |
| ch16rgb-4-3 | 216 | 110 | 244 | 82 | 267 | 59 | 236 | 90 | 696 | 282 | 0.7117 |
| ch16rgb-4-4 | 207 | 119 | 233 | 93 | 234 | 92 | 259 | 67 | 674 | 304 | 0.6892 |
| **Total** | | | | | | | | | **2778** | **1134** | **0.7101** |

Full Image, 4-way, 8-bin RGB Color Histogram

| Testing Set / Training Set | ch08rgb-4-1 | | ch08rgb-4-2 | | ch08rgb-4-3 | | ch08rgb-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | C | I | C | I | C | I | C | I | C | I | A |
| ch08rgb-4-1 | 248 | 78 | 234 | 92 | 239 | 87 | 228 | 98 | 701 | 277 | 0.7168 |
| ch08rgb-4-2 | 214 | 112 | 265 | 61 | 247 | 79 | 230 | 96 | 691 | 287 | 0.7065 |
| ch08rgb-4-3 | 220 | 106 | 237 | 89 | 264 | 62 | 244 | 82 | 701 | 277 | 0.7168 |
| ch08rgb-4-4 | 216 | 110 | 237 | 89 | 246 | 80 | 261 | 65 | 699 | 279 | 0.7147 |
| **Total** | | | | | | | | | **2792** | **1120** | **0.7137** |

Note: Self test results are omitted from the accuracy calculation.

## APPENDIX C – FULL IMAGE INDIVIDUAL CLASSIFIER RESULTS

C = Correct, I = Incorrect, A = Accuracy

Full Image, 4-way, 8-bin Lst Edge Histogram

| Testing Set | eh08lst-4-1 | | eh08lst-4-2 | | eh08lst-4-3 | | eh08lst-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| eh08lst-4-1 | 225 | 101 | 229 | 97 | 224 | 102 | 232 | 94 | 685 | 293 | 0.7004 |
| eh08lst-4-2 | 223 | 103 | 228 | 98 | 219 | 107 | 231 | 95 | 673 | 305 | 0.6881 |
| eh08lst-4-3 | 232 | 94 | 225 | 101 | 228 | 98 | 229 | 326 | 686 | 521 | 0.5684 |
| eh08lst-4-4 | 229 | 97 | 226 | 100 | 227 | 99 | 233 | 93 | 682 | 296 | 0.6973 |
| **Total** | | | | | | | | | **2726** | **1415** | **0.6583** |

Full Image, 4-way, 8-bin RGB Edge Histogram

| Testing Set | eh08rgb-4-1 | | eh08rgb-4-2 | | eh08rgb-4-3 | | eh08rgb-4-4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| eh08rgb-4-1 | 237 | 89 | 237 | 89 | 235 | 91 | 241 | 85 | 713 | 265 | 0.7290 |
| eh08rgb-4-2 | 240 | 86 | 239 | 87 | 228 | 98 | 238 | 88 | 706 | 272 | 0.7219 |
| eh08rgb-4-3 | 239 | 87 | 234 | 92 | 236 | 90 | 240 | 86 | 713 | 265 | 0.7290 |
| eh08rgb-4-4 | 238 | 88 | 233 | 93 | 232 | 94 | 241 | 85 | 703 | 275 | 0.7188 |
| **Total** | | | | | | | | | **2835** | **1077** | **0.7247** |

Full Image, 2-way

| | Trained on Set 1 | | Trained on Set 2 | | Total | | |
|---|---|---|---|---|---|---|---|
| Feature | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect | Accuracy |
| Color Histogram 8-bin Lst | 471 | 181 | 468 | 184 | 939 | 365 | **0.7201** |
| Color Histogram 8-bin RGB | 523 | 129 | 531 | 121 | 1054 | 250 | **0.8083** |
| Edge Histogram 8-bin RGB | 484 | 168 | 473 | 179 | 957 | 347 | **0.7339** |

Note: Self test results are omitted from the accuracy calculation.

## APPENDIX D – FULL IMAGE COMBINED CLASSIFIER RESULTS

These tables describe the accuracy of the indoor/outdoor image classification application using all three classifiers applied to the full image. The results from the color histogram, color coherence, and edge histogram classifiers are combined using three different strategies to produce the final result.

Summary: Combined Classifier Performance, 4-way

| | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 781 | 197 | 816 | 162 | 792 | 186 |
| Set 2 | 790 | 188 | 799 | 179 | 784 | 194 |
| Set 3 | 784 | 194 | 806 | 172 | 804 | 174 |
| Set 4 | 1061 | 243 | 1084 | 220 | 1042 | 262 |
| Total | 3416 | 822 | 3505 | 733 | 3422 | 816 |
| **Accuracy** | | **0.8060** | | **0.8270** | | **0.8075** |

Totals for this table were obtained using multiple test runs, available on the next page.

Summary: Combined Classifier Performance, 2-way

| | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 528 | 124 | 541 | 111 | 547 | 105 |
| Set 2 | 534 | 118 | 549 | 103 | 541 | 111 |
| Total | 1062 | 242 | 1090 | 214 | 1088 | 216 |
| **Accuracy** | | **0.8144** | | **0.8359** | | **0.8344** |

Note: Self test results are omitted from the accuracy calculation.

## APPENDIX D – FULL IMAGE COMBINED CLASSIFIER RESULTS

Detail: Combined Classifier Performance, 4-way, Trained on Set 4-1

| Test Set | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1* | 269 | 27 | 283 | 43 | 296 | 30 |
| Set 2 | 256 | 70 | 270 | 56 | 264 | 62 |
| Set 3 | 262 | 64 | 274 | 52 | 264 | 62 |
| Set 4 | 263 | 63 | 272 | 54 | 264 | 62 |
| Total | 781 | 197 | 816 | 162 | 792 | 186 |

Detail: Combined Classifier Performance, 4-way, Trained on Set 4-2

| Test Set | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 263 | 63 | 263 | 63 | 257 | 69 |
| Set 2* | 274 | 52 | 274 | 52 | 295 | 31 |
| Set 3 | 264 | 62 | 271 | 55 | 265 | 61 |
| Set 4 | 263 | 63 | 265 | 61 | 262 | 64 |
| Total | 790 | 188 | 799 | 179 | 784 | 194 |

Detail: Combined Classifier Performance, 4-way, Trained on Set 4-3

| Test Set | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 264 | 62 | 265 | 61 | 262 | 64 |
| Set 2 | 254 | 72 | 264 | 62 | 272 | 54 |
| Set 3* | 273 | 53 | 282 | 44 | 292 | 34 |
| Set 4 | 266 | 60 | 277 | 49 | 270 | 56 |
| Total | 784 | 194 | 806 | 172 | 804 | 174 |

Detail: Combined Classifier Performance, 4-way, Trained on Set 4-4

| Test Set | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 256 | 70 | 264 | 62 | 253 | 73 |
| Set 2 | 260 | 66 | 269 | 57 | 249 | 77 |
| Set 3 | 268 | 58 | 268 | 58 | 264 | 62 |
| Set 4* | 277 | 49 | 283 | 43 | 276 | 50 |
| Total | 1061 | 243 | 1084 | 220 | 1042 | 262 |

(*) Note: Self test results are omitted from the accuracy calculation.

## APPENDIX E – SUBBLOCKED IMAGE INDIVIDUAL CLASSIFIER RESULTS

These tables describe the classification accuracy of the individual classifiers operating on the segmented images.

Subblocked Image, 4-way, 8-bin Lst Color Histogram

| Testing Set | Set 1 | | Set 2 | | Set 3 | | Set 4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| Set 1 | 3581 | 1635 | 3342 | 1874 | 3398 | 1818 | 3439 | 1777 | 10179 | 5469 | 0.6505 |
| Set 2 | 3326 | 1890 | 3317 | 1899 | 3626 | 1590 | 3428 | 1768 | 10380 | 5268 | 0.6633 |
| Total | | | | | | | | | 20559 | 10737 | **0.6569** |

Subblocked Image, 4-way, 8-bin RGB Color Coherence

| Testing Set | Set 1 | | Set 2 | | Set 3 | | Set 4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| Set 1 | 3932 | 1284 | 2780 | 1436 | 3810 | 1406 | 3789 | 1427 | 10379 | 4269 | 0.7086 |
| Set 2 | 3714 | 1502 | 3980 | 1236 | 3632 | 1384 | 3749 | 1467 | 11295 | 4353 | 0.7216 |
| Total | | | | | | | | | 21674 | 8622 | **0.7154** |

Subblocked Image, 4-way, 8-bin RGB Edge Histogram

| Testing Set | Set 1 | | Set 2 | | Set 3 | | Set 4 | | Total | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Training Set | C | I | C | I | C | I | C | I | C | I | A |
| Set 1 | 3342 | 1874 | 3339 | 1877 | 3299 | 1917 | 3364 | 1852 | 10002 | 5646 | 0.6392 |
| Set 2 | 3339 | 1877 | 3344 | 1872 | 3296 | 1920 | 3331 | 1865 | 9966 | 5682 | 0.6369 |
| Total | | | | | | | | | 19968 | 11328 | **0.6380** |

Subblocked Image, 2-way

| | Trained on Set 1 | | Trained on Set 2 | | Total | | |
|---|---|---|---|---|---|---|---|
| Feature | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect | Accuracy |
| Color Histogram 6-bin Lst | 520 | 132 | 496 | 156 | 1016 | 288 | **0.7791** |
| Color Histogram 8-bin RGB | 558 | 94 | 554 | 98 | 1112 | 192 | **0.8528** |
| Edge Histogram 8-bin RGB | 509 | 143 | 509 | 143 | 1018 | 286 | **0.7807** |

Note: Self test results are omitted from the accuracy calculation.

## APPENDIX F – SUBBLOCKED IMAGE COMBINED CLASSIFIER RESULTS

These tables describe the final accuracy of the indoor/outdoor image classification application. Each image is segmented into 16 blocks, which are classified using color histogram, color coherence, and edge histogram features. The 16 features are summed to produce a feature classifier judgment, and the final classification is obtained by three different strategies.

Summary: Combined Classifier Performance, 4-way

| | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 835 | 143 | 844 | 134 | 829 | 149 |
| Set 2 | 842 | 136 | 850 | 128 | 855 | 123 |
| Total | 1677 | 279 | 1694 | 262 | 1684 | 272 |
| **Accuracy** | | **0.8574** | | **0.8661** | | **0.8609** |

Totals for this table were obtained using multiple test runs, available on the next page. Sets 3 and 4 were not trained.

Summary: Combined Classifier Performance, 2-way

| | Majority | | Distance | | Second Stage SVM | |
|---|---|---|---|---|---|---|
| | Correct | Incorrect | Correct | Incorrect | Correct | Incorrect |
| Set 1 | 567 | 85 | 569 | 83 | 575 | 77 |
| Set 2 | 558 | 94 | 568 | 84 | 571 | 81 |
| Total | 1125 | 179 | 1137 | 167 | 1146 | 158 |
| **Accuracy** | | **0.8627** | | **0.8719** | | **0.8788** |

Note: Self test results are omitted from the accuracy calculation.

## APPENDIX G – IMAGE FEATURE EXTRACTION PERFORMANCE

This table lists the performance of the feature extractor running on one processor, on the cluster using a cached network file system, and on the cluster using a custom input/output communications system.

Parallel image classification feature extractor, 10 processors

| Testing Set | | Sequential | | Parallel with NFS Cache | | | Parallel with Custom IO | | |
|---|---|---|---|---|---|---|---|---|---|
| Set | Images | Time | Images/s | Time | Images/s | Speedup | Time | Images/s | Speedup |
| list 4_1 | 326 | 143.30 | 2.28 | 23.71 | 13.75 | 6.04 | 53.59 | 6.08 | 2.67 |
| list 4_2 | 326 | 141.31 | 2.31 | 23.83 | 13.68 | 5.93 | 54.60 | 5.97 | 2.59 |
| list 4_3 | 326 | 144.18 | 2.26 | 23.87 | 13.66 | 6.04 | 55.42 | 5.88 | 2.60 |
| list 4_4 | 326 | 140.79 | 2.32 | 27.97 | 11.65 | 5.03 | 55.11 | 5.92 | 2.55 |
| list 2a_1 | 652 | 290.78 | 2.24 | 60.00 | 10.87 | 4.85 | 110.55 | 5.90 | 2.63 |
| list 2a_2 | 652 | 259.65 | 2.51 | 53.29 | 12.23 | 4.87 | 108.36 | 6.02 | 2.40 |
| **Average** | | | **2.32** | | **12.64** | **5.46** | | **5.96** | **2.57** |

## APPENDIX H – IMAGE CLASSIFICATION OUTPUT EXAMPLE

This execution output demonstrates the feedback provided by the image classification software for one of the final test runs.

```
[matthew@cluster parallel]$ mpirun -np 10 -machinefile cluster ./fastextract \
                          -r 2 -c c17-2a-1.cfg -i ../test/list_2a_2.txt
[--]                : Welcome to the Parallel Feature Extractor
[--]                : Matthew Woitaszek
[--]                : 28 April 2002
[--]                :
[ 0]CFS:Check       : Runmode:      2
[ 0]CFS:Check       : Input file:   ../test/list_2a_2.txt
[ 0]CFS:Check       : Image path:   ../images
[ 0]CFS:Check       : Output base:  list_2a_2
[ 0]CFS:Check       : CH Model:  ../test/models/c17-2a-1-ch-08lst-model
[ 0]CFS:Check       : CC Model:  ../test/models/c17-2a-1-cc-08rgb-model
[ 0]CFS:Check       : EH Model:  ../test/models/c17-2a-1-eh-08rgb-model
[ 0]                : Assignment to Color Histogram complete (status = 2)
[ 0]                : Done on cluster
[--]                : Elapsed Time = 3037.271486 seconds
[--]                : Processing complete.


[ 1]                : Assignment to Color Coherence complete (status = 2)
[ 1]                : Done on a

[ 8]                : Assignment to Edge Histogram complete (status = 2)
[ 8]                : Done on j

[ 3]                : Assignment to Color Coherence complete (status = 2)
[ 3]                : Done on e

[ 4]                : Assignment to Color Coherence complete (status = 2)
[ 4]                : Done on f

[ 7]                : Assignment to Edge Histogram complete (status = 2)
[ 7]                : Done on i

[ 2]                : Assignment to Color Coherence complete (status = 2)
[ 2]                : Done on b

[ 6]                : Assignment to Edge Histogram complete (status   2)
[ 6]                : Done on h

[ 5]                : Assignment to Edge Histogram complete (status   2)
[ 5]                : Done on g

[ 9]Collector:Prep : Reading CH Model...
[ 9]CFS:ReadModel   :   Reading model file
[ 9]CFS:ReadModel   :   parameter_C=10.000000
[ 9]CFS:ReadModel   :   parameter_e=0.001000
[ 9]CFS:ReadModel   :   parameter_tol=0.001000
[ 9]CFS:ReadModel   :   parameter_b=-0.086165
[ 9]CFS:ReadModel   :   kernel_type=1
[ 9]CFS:ReadModel   :   kernel_rbf_sigma=1.000000
[ 9]CFS:ReadModel   :   Read 25 weight vectors
[ 9]CFS:ReadModel   :   Read 7587 support vectors
[ 9]CFS:ReadModel   :   Model read complete
[ 9]Collector:Prep :   Model = 7587 plus 16 blanks 7603 total
[ 9]Collector:Prep : Reading CC Model...
[ 9]CFS:ReadModel   :   Reading model file
[ 9]CFS:ReadModel   :   parameter_C=10.000000
[ 9]CFS:ReadModel   :   parameter_e=0.001000
[ 9]CFS:ReadModel   :   parameter_tol=0.001000
[ 9]CFS:ReadModel   :   parameter_b=3.108036
[ 9]CFS:ReadModel   :   kernel_type=1
[ 9]CFS:ReadModel   :   kernel_rbf_sigma=1.000000
```

```
[ 9]CFS:ReadModel  :   Read 25 weight vectors
[ 9]CFS:ReadModel  :   Read 6398 support vectors
[ 9]CFS:ReadModel  :   Model read complete
[ 9]Collector:Prep :   Model = 6398 plus 16 blanks 6414 total
[ 9]Collector:Prep : Reading EH Model...
[ 9]CFS:ReadModel  :   Reading model file
[ 9]CFS:ReadModel  :   parameter_C=10.000000
[ 9]CFS:ReadModel  :   parameter_e=0.001000
[ 9]CFS:ReadModel  :   parameter_tol=0.001000
[ 9]CFS:ReadModel  :   parameter_b=8.339960
[ 9]CFS:ReadModel  :   kernel_type=1
[ 9]CFS:ReadModel  :   kernel_rbf_sigma=1.000000
[ 9]CFS:ReadModel  :   Read 25 weight vectors
[ 9]CFS:ReadModel  :   Read 8376 support vectors
[ 9]CFS:ReadModel  :   Model read complete
[ 9]Collector:Prep :   Model = 8376 plus 16 blanks 8392 total
[ 9]Collector:Prep : Reading S2 Model...
[ 9]CFS:ReadModel  :   Reading model file
[ 9]CFS:ReadModel  :   parameter_C=10.000000
[ 9]CFS:ReadModel  :   parameter_e=0.001000
[ 9]CFS:ReadModel  :   parameter_tol=0.001000
[ 9]CFS:ReadModel  :   parameter_b=0.120586
[ 9]CFS:ReadModel  :   kernel_type=1
[ 9]CFS:ReadModel  :   kernel_rbf_sigma=1.000000
[ 9]CFS:ReadModel  :   Read 4 weight vectors
[ 9]CFS:ReadModel  :   Read 171 support vectors
[ 9]CFS:ReadModel  :   Model read complete
[ 9]Collector:Prep :   Model = 171 plus 1 blanks 172 total
[ 9]               : Assignment to Collector complete (status = 2)

Individual Classifier Performance

    ch  cc  eh         count
 0  .   .   .            24 (0.037)
 1  .   .   +            27 (0.041)
 2  .   +   .            14 (0.021)
 3  .   +   +            67 (0.103)
 4  +   .   .            20 (0.031)
 5  +   .   +            23 (0.035)
 6  +   +   .            85 (0.130)
 7  +   +   +           392 (0.601)

Results Summary

                       correct           incorrect
Color Histogram Only       520 (0.798)        132 (0.202)
Color Coherence Only       558 (0.856)         94 (0.144)
Edge Histogram Only        509 (0.781)        143 (0.219)

Majority Classifier        567 (0.870)         85 (0.130)
Distance Classifier        569 (0.873)         83 (0.127)
Second Stage SV Machine    575 (0.882)         77 (0.118)


[ 9]               : Done on k
```

## APPENDIX I – MAIL CLASSIFICATION TEST SETS

This table describes the data sets used in the electronic mail classification application, including the number of original messages in the data set and the number of messages with sufficient text for successful parsing by the feature extractor. Messages which contain no dictionary words are omitted.

Data Set Size and Feature Extraction Omission Rate with Personal Dictionary

| | Original Messages | | | Parsed Messages | | Omitted Messages | | Omission |
|---|---|---|---|---|---|---|---|---|
| Data Set | Total | Spam | Nonspam | Spam | Nonspam | Spam | Nonspam | Rate |
| Set A Training | 1340 | 670 | 670 | 666 | 656 | 4 | 14 | **1.34%** |
| Set A Testing | 1344 | 672 | 672 | 668 | 663 | 4 | 9 | **0.97%** |
| Set B All SPAM | 77 | 77 | 0 | 75 | 0 | 2 | 0 | **2.60%** |
| Set C | 378 | 0 | 378 | 0 | 348 | 0 | 30 | **7.94%** |

# APPENDIX J – MAIL CLASSIFICATION RESULTS

## Personalized Dictionary

| Set | Total Examples | | Correctly Classified | | Incorrectly Classified | | Accuracy |
|---|---|---|---|---|---|---|---|
| | Spam | Nonspam | As Spam | As Nonspam | As Spam | As Nonspam | |
| Set A Testing | 668 | 663 | 654 | 633 | 30 | 14 | **0.9669** |
| Set B All Spam | 75 | 0 | 73 | 0 | 0 | 2 | **0.9733** |
| Set C Subset | 0 | 348 | 0 | 335 | 13 | 0 | **0.9626** |
| Set C Complete | 0 | 358 | 0 | 335 | 23 | 0 | **0.9358** |

## Impersonal Dictionary

| Set | Total Examples | | Correctly Classified | | Incorrectly Classified | | Accuracy |
|---|---|---|---|---|---|---|---|
| | Spam | Nonspam | As Spam | As Nonspam | As Spam | As Nonspam | |
| Set A Testing | 668 | 663 | 654 | 633 | 30 | 14 | **0.9669** |
| Set B All Spam | 75 | 0 | 72 | 0 | 0 | 3 | **0.9600** |
| Set C Subset | 0 | 348 | 0 | 336 | 12 | 0 | **0.9655** |
| Set C Complete | 0 | 358 | 0 | 336 | 22 | 0 | **0.9385** |

## System Dictionary

| Set | Total Examples | | Correctly Classified | | Incorrectly Classified | | Accuracy |
|---|---|---|---|---|---|---|---|
| | Spam | Nonspam | As Spam | As Nonspam | As Spam | As Nonspam | |
| Set A Testing | 666 | 662 | 648 | 617 | 45 | 18 | **0.9526** |
| Set B All Spam | 75 | 0 | 67 | 0 | 0 | 8 | **0.8933** |
| Set C Subset | 0 | 348 | 0 | 324 | 24 | 0 | **0.9310** |
| Set C Complete | 0 | 358 | 0 | 324 | 34 | 0 | **0.9050** |

## APPENDIX K – MAIL CLASSIFICATION TRAINED DICTIONARY MODELS

These tables lists some of the most important words considered when classifying an electronic mail message as spam or legitimate. Both were obtained by training the feature dictionary with a linear SVM which produces a weight for each possible word.

Personal Dictionary: Contains proper nouns related to recipients collecting samples

| **Words Indicating Nonspam Messages** | | | | **Words Indicating Spam Messages** | | | |
|---|---|---|---|---|---|---|---|
| Word | Weight | Word | Weight | Word | Weight | Word | Weight |
| **matthew** | **-0.4070** | what | -0.12809 | subject | 0.1328 | adult | 0.1791 |
| i | -0.3666 | **rit** | **-0.12686** | business | 0.1342 | their | 0.1796 |
| do | -0.2943 | **woitaszek** | **-0.12584** | remove | 0.1391 | at | 0.1965 |
| read | -0.2663 | attached | -0.12308 | receive | 0.1406 | these | 0.1978 |
| go | -0.2531 | always | -0.12093 | our | 0.1429 | must | 0.2021 |
| mom | -0.2140 | heres | -0.11969 | action | 0.1432 | hot | 0.2040 |
| e | -0.1868 | thanks | -0.11695 | offers | 0.1436 | be | 0.2090 |
| would | -0.1839 | almost | -0.11403 | incredible | 0.1456 | sex | 0.2103 |
| dad | -0.1708 | some | -0.11375 | from | 0.1459 | shipping | 0.2134 |
| think | -0.1690 | but | -0.10901 | guaranteed | 0.1503 | porn | 0.2317 |
| day | -0.1629 | between | -0.10874 | access | 0.1507 | more | 0.2343 |
| **brost** | **-0.1562** | when | -0.10856 | and | 0.1538 | sites | 0.2346 |
| web | -0.1548 | he | -0.10762 | want | 0.1548 | removed | 0.2428 |
| the | -0.1543 | somebody | -0.10703 | no | 0.1551 | celebs | 0.2524 |
| how | -0.1489 | talk | -0.10697 | now | 0.1581 | best | 0.2550 |
| page | -0.1486 | new | -0.10613 | list | 0.1602 | unsubscribe | 0.2573 |
| will | -0.1409 | then | -0.10484 | show | 0.1613 | to | 0.2597 |
| that | -0.1408 | ny | -0.10469 | take | 0.1648 | /snip/ | 0.2888 |
| documentation | -0.1406 | its | -0.10456 | very | 0.1661 | tight | 0.2888 |
| **rochester** | **-0.1406** | mail | -0.10436 | below | 0.1689 | click | 0.3241 |
| kroc | -0.1361 | good | -0.10359 | money | 0.1691 | teens | 0.3430 |
| am | -0.1351 | lk | -0.10347 | get | 0.1709 | hardcore | 0.3650 |
| needs | -0.1348 | temp | -0.10347 | future | 0.1729 | young | 0.3932 |
| msn | -0.1338 | hope | -0.10184 | mailings | 0.1734 | here | 0.4521 |
| night | -0.1289 | are | -0.10178 | handle | 0.1747 | free | 0.5159 |

## APPENDIX K – MAIL CLASSIFICATION TRAINED DICTIONARY MODELS

Impersonal Dictionary: References to 14 proper nouns related to recipients removed

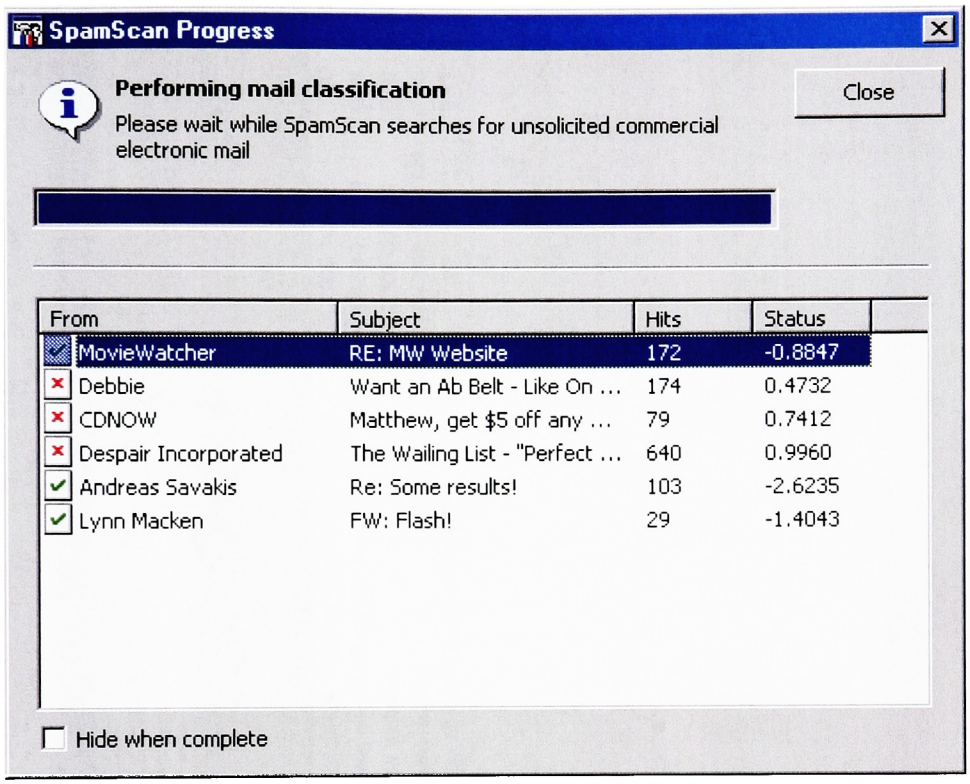
| Words Indicating Nonspam Messages | | | | Words Indicating Spam Messages | | | |
|---|---|---|---|---|---|---|---|
| Word | Weight | Word | Weight | Word | Weight | Word | Weight |
| i | -0.3565 | its | -0.1230 | available | 0.1464 | mailings | 0.2008 |
| do | -0.3038 | hello | -0.1222 | business | 0.1465 | these | 0.2052 |
| e | -0.2642 | how | -0.1217 | mailing | 0.1522 | future | 0.2096 |
| read | -0.2453 | surprise | -0.1215 | action | 0.1523 | hot | 0.2164 |
| go | -0.2428 | what | -0.1213 | watch | 0.1528 | shipping | 0.2172 |
| would | -0.2045 | he | -0.1195 | from | 0.1542 | adult | 0.2172 |
| that | -0.1858 | still | -0.1184 | remove | 0.1563 | at | 0.2189 |
| mom | -0.1785 | think | -0.1183 | offers | 0.1568 | sex | 0.2309 |
| pm | -0.1749 | thanks | -0.1179 | internet | 0.1578 | must | 0.2329 |
| hey | -0.1698 | am | -0.1155 | money | 0.1613 | be | 0.2332 |
| day | -0.1681 | night | -0.1153 | want | 0.1647 | porn | 0.2474 |
| the | -0.1605 | needs | -0.1141 | access | 0.1681 | more | 0.2528 |
| page | -0.1520 | always | -0.1138 | incredible | 0.1714 | sites | 0.2556 |
| msn | -0.1479 | hope | -0.1134 | list | 0.1728 | celebs | 0.2706 |
| may | -0.1471 | almost | -0.1131 | to | 0.1792 | best | 0.2718 |
| when | -0.1347 | will | -0.1129 | their | 0.1823 | removed | 0.2731 |
| web | -0.1327 | as | -0.1121 | guaranteed | 0.1828 | /snip/ | 0.3003 |
| hi | -0.1308 | documentation | -0.1120 | very | 0.1837 | tight | 0.3003 |
| then | -0.1303 | know | -0.1108 | take | 0.1861 | unsubscribe | 0.3032 |
| im | -0.1290 | note | -0.1087 | get | 0.1874 | click | 0.3148 |
| thursday | -0.1283 | virtual | -0.1084 | below | 0.1874 | teens | 0.3563 |
| dad | -0.1275 | questions | -0.1080 | no | 0.1888 | hardcore | 0.3907 |
| are | -0.1251 | some | -0.1070 | handle | 0.1888 | young | 0.4131 |
| but | -0.1246 | ny | -0.1067 | show | 0.1889 | here | 0.4850 |
| course | -0.1236 | perhaps | -0.1042 | now | 0.1910 | free | 0.5452 |

## APPENDIX K – MAIL CLASSIFICATION TRAINED DICTIONARY MODELS

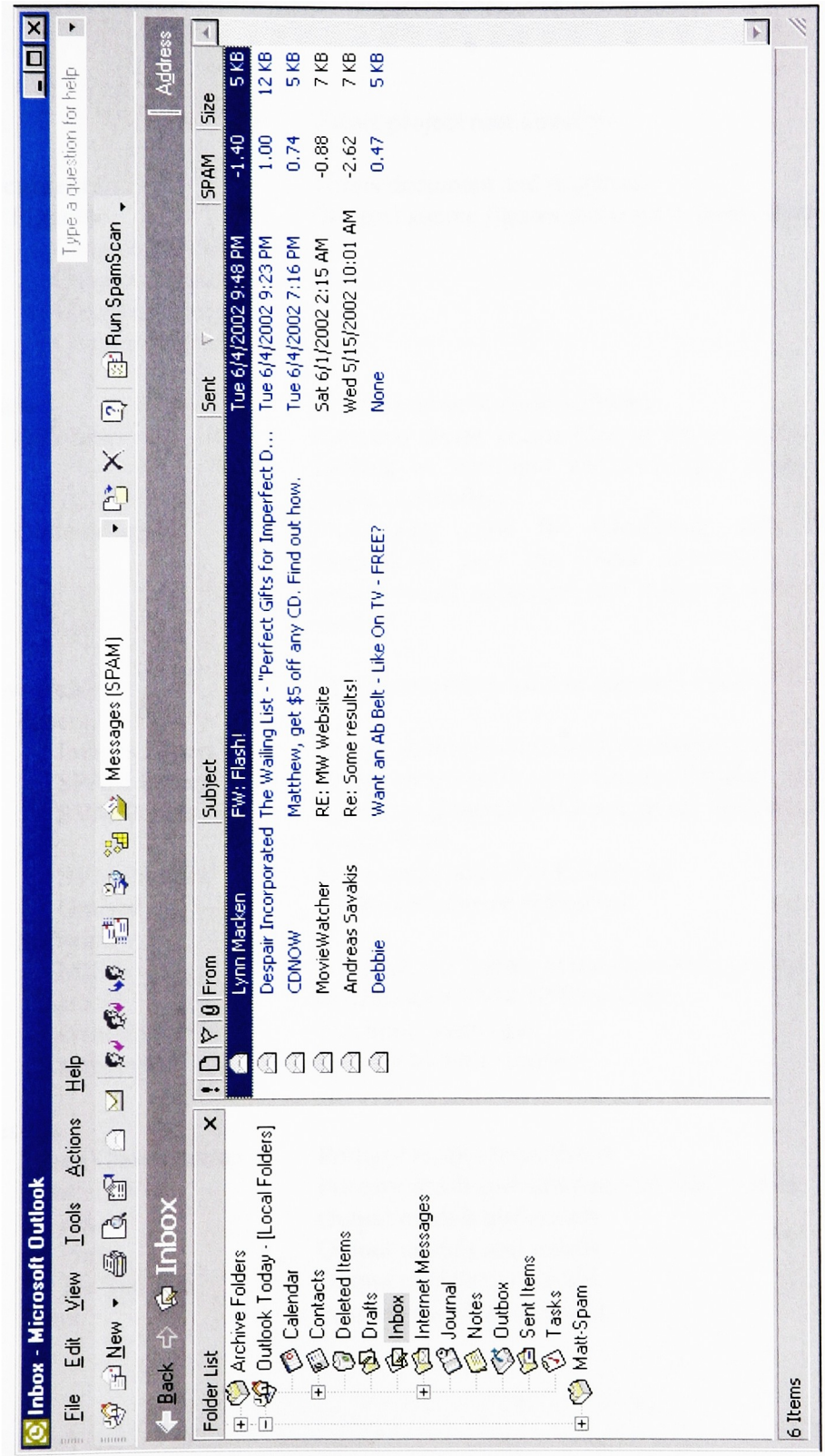
System Dictionary: Trained with wordlist from /usr/dict

| Words Indicating Nonspam Messages | | | | Words Indicating Spam Messages | | | |
|---|---|---|---|---|---|---|---|
| Word | Weight | Word | Weight | Word | Weight | Word | Weight |
| i | -0.5416 | virtual | -0.1986 | take | 0.1721 | line | 0.2695 |
| do | -0.4877 | technology | -0.1986 | because | 0.1734 | money | 0.2790 |
| read | -0.3626 | speak | -0.1981 | very | 0.1739 | receive | 0.2791 |
| hi | -0.3593 | page | -0.1963 | secret | 0.1768 | show | 0.2815 |
| go | -0.3372 | night | -0.1910 | business | 0.1799 | march | 0.2836 |
| e | -0.2952 | well | -0.1907 | antenna | 0.1840 | watch | 0.2899 |
| day | -0.2931 | know | -0.1875 | get | 0.1868 | more | 0.2907 |
| hope | -0.2925 | mail | -0.1840 | want | 0.1888 | handle | 0.3037 |
| group | -0.2873 | prevention | -0.1815 | at | 0.1892 | list | 0.3231 |
| the | -0.2627 | were | -0.1814 | this | 0.1894 | mar | 0.3325 |
| mike | -0.2595 | always | -0.1801 | to | 0.2011 | best | 0.3464 |
| hey | -0.2496 | may | -0.1749 | my | 0.2025 | be | 0.3509 |
| would | -0.2492 | he | -0.1749 | incredible | 0.2036 | no | 0.3589 |
| will | -0.2420 | save | -0.1708 | subject | 0.2080 | future | 0.3614 |
| still | -0.2383 | let | -0.1688 | find | 0.2091 | /snip/ | 0.3626 |
| but | -0.2376 | course | -0.1663 | hot | 0.2341 | tight | 0.3626 |
| almost | -0.2301 | note | -0.1646 | date | 0.2387 | font | 0.3679 |
| original | -0.2258 | how | -0.1637 | below | 0.2504 | sex | 0.3804 |
| that | -0.2190 | documentation | -0.1632 | our | 0.2504 | privacy | 0.3821 |
| am | -0.2078 | sure | -0.1594 | adult | 0.2525 | body | 0.3874 |
| sent | -0.2050 | as | -0.1590 | must | 0.2559 | click | 0.4039 |
| are | -0.2039 | mark | -0.1583 | life | 0.2561 | these | 0.4999 |
| then | -0.2012 | web | -0.1573 | off | 0.2630 | here | 0.6207 |
| surprise | -0.2005 | talk | -0.1544 | their | 0.2670 | young | 0.6516 |
| when | -0.2002 | hello | -0.1542 | now | 0.2689 | free | 0.6791 |

# APPENDIX L – MAIL CLASSIFICATION INTERFACE



# APPENDIX M – MAIL CLASSIFICATION OUTLOOK INTEGRATION



## APPENDIX N – CD-ROM CONTENT LIST

| | |
|---|---|
| **Thesis** | Thesis project root directory |
| **Document** | Thesis document and proposal |
| Figures | Original source figures included in thesis document |
| Chapter 2 SVM | |
| Chapter 3 Parallel | |
| Chapter 4 Image | |
| Chapter 5 SPAM | |
| **Misc** | Miscellaneous support software |
| DSTConv | Converts dense vector files to sparse vector files for training in svmLight and svmLight model files to msvm model files |
| OutlookTest | Preliminary code for interfacing with Microsoft Outlook to view the folder hierarchy, access and export e-mail messages, and integrate with the object model |
| **Research** | Original research used in literature review |
| Papers | |
| Images Papers | Papers on image classification: Serrano, Szummer |
| SPAM Papers | Papers on spam filtering: Cranor, Drucker, Sahami |
| SVM Papers | Papers on general SVM operation: Burges, Joachims, Osuna, Platt |
| SVM Parallel | Paper on parallel SVM: Schmuel |
| Unused | Additional unused references |
| Software | |
| MIST | Sue Muller's feature extraction thesis project |
| smobr | deAlmeida's SVM-SMO trainer |
| svmlight | Joachims' svmLight |
| svm-smo | Ge's SVM-SMO trainer |
| **Results** | |
| Image Classification | Primary result spreadsheets |
| Parallel SVM | Primary result spreadsheets and shell scripts |
| Adult | Output models and results |
| Pima | Output models and results |
| Tic Tac Toe | Output models and results |
| Spam | Primary result spreadsheet |

## APPENDIX N – CD-ROM CONTENT LIST

| | |
|---|---|
| **Source** | Hierarchy moved from Linux cluster |
| **image** | Image classification application |
| images | Image files |
| parallel | **fastextract**: parallel extractor and classifier |
| sequential | **fastextract**: sequential extractor |
| test | Test sets |
| entireimage | Models used by test sets |
| models | Models used by test sets |
| **msvm** | Support Vector Machine trainer and classifier |
| blocked | **msvm**: blocked implementation |
| interleaved | **msvm**: interleaved implementation |
| sequential | **msvm**: sequential implementation |
| test | Original test sets |
| **spam** | E-mail classification application |
| data | Spam data files |
| tokenizer | **tokenizer**: dictionary and feature vector generator<br>**spamscan**: uses dictionary.txt and model.txt to generate classifications from standard input |
| **SpamScan** | Microsoft Outlook XP e-mail classification add-in |
| Resource | Bitmap resources |